工學博士學位論文

저전력 프로세서 캐시를 위한 구조적 개선 방안

Architectural Improvements for Low-power Processor Cache

2013年 7月

서울대학교 대학원

전기·컴퓨터 공학부

조 윤 교

요 약

마이크로프로세서는 수행 성능을 증가시키고 소모하는 에너지를 줄이기 위해 연구가 진행되고 있다. 대부분의 경우 수행 성능과 소모 에너지들 간에는 트레이드오프(trade-off) 관계가 성립하여, 소모 에너지를 감소시키면 수행 성능이 낮아지게 된다. 본 논문에서는 프로세서의 구조적 개선을 통해, 수행 성능에 영향을 미치지 않으면서 소모 에너지를 감소시키는 방안과 수행 성능에 큰 영향을 미치는 여러 에너지 감소 방안들을 오버헤드를 최소화하면서 조합하는 방안을 제안한다.

첫 번째로, 수행 성능에 영향을 미치지 않으며 동적 에너지를 감소시키기 위해 '선택적 워드 접근 기법'을 제안한다. 저장장치 별 저장 단위가 다르다는 점에 착안한 이 기법은 주소의 일부분을 캐시 접근 시에 활용하여 저장장치 별로 필요한 부분만을 전달한다. 이 기법을 모의 실험하여 L1 캐시에서 67.5%, L2 캐시에서 27.1%의 동적 에너지 감소를 이끌어 냈다. 정적 에너지까지 고려하면 L1 캐시에서 56.75%의 에너지 감소를 이끌어 냈다.

두 번째로, 수행 성능에 큰 영향을 미치는 필터 캐시, 순차적 캐시 그리고 드라우지 캐시와 논문 전반부에서 제시한 선택적 워드 접근 기법을, 오버헤드를 최소화하면서 조합하는 '워드 필터를 사용한 순차적, 선택적 워드 접근 드라우지 캐시'를 제안한다.

필터 캐시는 프로세서 레지스터와 L1 캐시 사이에 작은 저장장치를 구현하여 동적 에너지 소모량을 줄이는 기법이다. 해당 기법이 처음 제시되었을 때와 달리 클록 수의 증가로 인해 L1 캐시 접근 시간이 늘어나고, 이로 인해 필터 캐시를 사용할 경우 에너지의 감소와 함께 성능상의 이득까지 볼 수 있다. 이와 함께 기존에 성능상의 손해로 인해 쓰지 못했던 순차적 캐시와 드라우지 캐시와 같은 기법들을 추가적으로 사용할 수 있다.

순차적 캐시는 캐시의 태그 어레이의 적중 여부를 알기 전까지 데이터 어레이를 동작시키지 않는 기법이다. 이는 태그 어레이의 적중 시간만큼 캐시 접근 시간이 길어지는 반면, 적중된 웨이만을 구동시키면 되기 때문에 데이터 어레이의 동적 에너지를 감소시킬 수 있다. 필터 캐시와 같이 사용할 경우, 상대적으로 전력 소모가 적은 태그 어레이를 필터 캐시와 병렬적으로 접근하게 되면 기존 순차적 캐시에서 손해를 보는 태그 어레이 접속 시간을 숨길 수 있다.

드라우지 캐시는 SRAM 셀에 동작전압을 정상 모드(높은 전압)와 저전력 모드(낮은 전압), 두 종류를 공급하고 동작이 발생하지 않는 부분의 전압을 낮추어 공급함으로 캐시의 정적 전력 소모를 줄이는 기법이다. 저전력 모드에 있는 셀에 접근할 경우 낮은 전압을 높은 전압으로 바꾸어주는데 이때 추가적인 접근 시간이 발생한다. 본 논문에서는 해당 셀에 접근하여 전압을 높이는 깨움 비트 전송을 필터 캐시와 L1 캐시 태그 어레이 접속과 병렬적으로 하여 기존 드라우지 캐시에서 발생하게 되는 성능 감소를 막았다.

이와 같이 드라우지 캐시 기법과, 필터 캐시, 순차적 캐시와 선택적 워드 접근 기법을 모두 적용하여 모의 실험한 결과, 전체 프로세서 캐시에서 73.4%의 동적 에너지 감소를, 83.2%의 정적 에너지 감소를, 총 71.7%의 에너지 감소를 이끌어 내었다.

요약하면, 정적 에너지 감소를 위해 드라우지 캐시를 구현하면서 발생하는 추가 시간을 필터 캐시와 순차적 캐시를 이용해 효율적으로 숨기고, 저장 단위 차이를 이용하는 선택적 워드 접근 기법을 추가적으로 구현해 저전력 프로세서 설계를 하였다.

주요어 : 선택적 워드 접근(Selective Word Reading), 필터 캐시(Filter Cache), 순차적 캐시(Sequential Cache), 드라우지 캐시(Drowsy Cache), 병렬적 접근 구조(Parallel Request Structure), 저전력 캐시 구조(Low-power Cache Architecture)

학번 : 2006-23201

목 차

그림 목차

표 목차

제 1장 서 론

1.1 연구 배경

1971년 미국 인텔사의 i4004로 시작된 마이크로프로세서 혁명 이후로, 2011년 Ivy Bridge에 이르기까지 반도체 구현 기술의 비약적 발전으로 속도는 4000배, 집적도는 36만 배 개선되었고, 고성능 컴퓨터 구조 기술 적용으로 IPC는 400배 향상, 단위 트랜지스터 당 전력 소모가 5만 배 개선되었다. 마이크로프로세서는 지난 40년간 성능/(가격x전력소모)가 총 3경배(3x1016)을 기록, 매년 2.58배씩 개선되어 왔고 그림 1.1과 같이 매달 8.2% 매주 2% 개선의 기술 혁명이 계속 진행 중이다.[KKS+10]

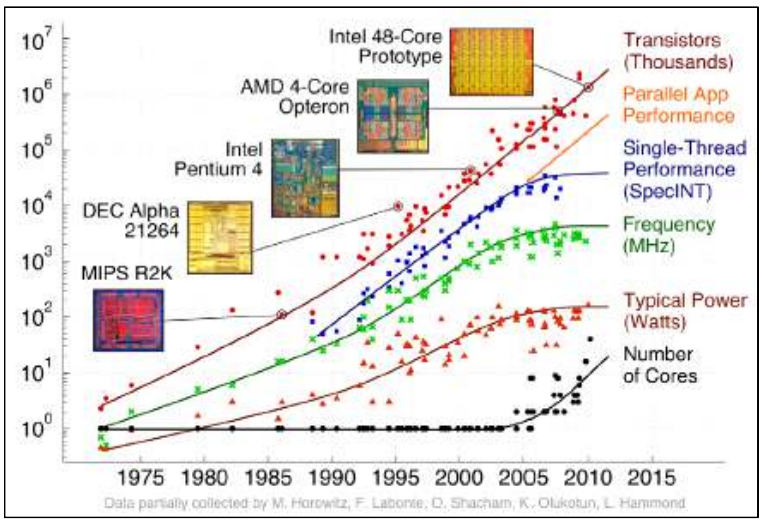

그림 1.1 마이크로프로세서 기술 발전 추세

그러나 2005년을 전후하여 단일 프로세서의 동작 주파수가 3GHz 내외에서 더 이상 개선되지 못하고 있는데 이것은 45nm 이하 디바이스에서 칩 당 전력소모가 패키지의 팬 냉각 한도인 200W로 도달하였기 때문이다. 이에 상대적으로 단순한 저전력 복수 개의 코어를 내장하여 멀티-코어를 통해 성능을 개선해나가는 대안이 구축되었고 이는 데스크탑, 수퍼 컴퓨터 뿐만 아니라 내장형 프로세서에도 적용되고 있다.

2000년대 후반에 들어서며 전력 소모는 시스템 설계에 가장 중요한 변수가 되었으며 특히 그림 1.2와 같이 수십만대의 서버를 거대한 건물 내에 집적하여 운용하는 수퍼 컴퓨터 혹은 클라우드 컴퓨팅 데이터 센터들은 1년간 전기료만 수천만 달러에 달하는 등 그린 수퍼 컴퓨팅의 요구가 극대화되고 있다.

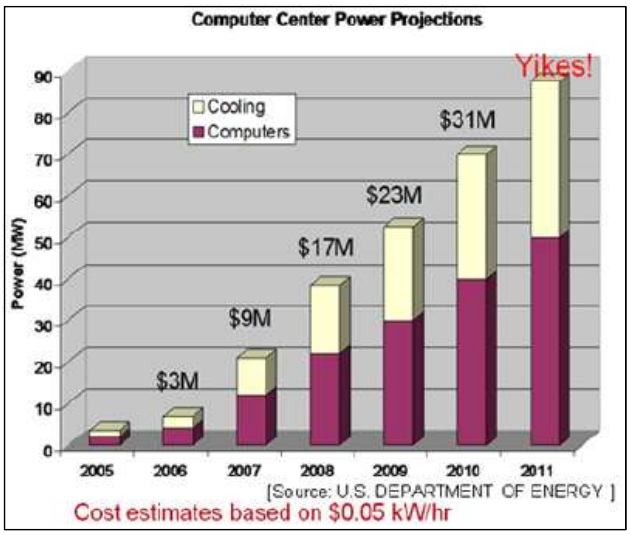

그림 1.2 Data Center 전력 소모-운용 비용

캐시 메모리는 프로세서와 메모리 사이의 속도 차이를 극복하기 위해 도입되었다. 시간적·공간적 연관성(temporal and spatial locality)을 도입하여 메모리 접근 시간을 줄이고 전체 시스템 향상에 큰 도움을 준다. 성능의 향상이 크기 때문에 자원적인 제약이 심한 내장형 프로세서에도 캐시 메모리의 활용도는 점점 높아지고 있다. 그런데, 점점 캐시의 용량이 커지고 구조가 복잡해짐에 따라 캐시가 전체 칩의 전력 소모에 미치는 영향도 커지고 있다. Alpha 21264에서 16%, StrongARM-110에서 43%, ARM-920T에서 44%의 전력 소모를 캐시 메모리가 차지하고 있으며, 2011년 INTEL에서 발표한 자료인 그림 1.3, 그림 1.4에 따르면 최신형 프로세서에서도 응용프로그램 종류에 따라 12%에서 45%까지의 전력 소모를 캐시 메모리가 차지하고 있는 것을 알 수 있다. [Soda11]

캐시 메모리의 전력 소모량은 정적 소모량과 동적 소모량으로 나눌 수 있다. 동적 소모량은 캐시에 실제 입출력이 발생할 때, 데이터를 읽어오는 과정에서 트린지스터가 스위칭(switching)되면서 발생하는 에너지를 말하고 정적 소모량은 캐시를 이루고 있는 SRAM의 각 트렌지스터 셀에 미량의 누설 전류(Leakage Current)가 흘러 발생하는 에너지를 말한다. 고성능화에 따른 미세 공정의 전력 관리에서 누설전력(leakage power)에 대한 비중은 점차 높아지고 있는 추세이며, 누설 전력은 공정기술이 진화함에 따라서 전체 프로세서 소비전력에서 상당부분을 차지하는 주요 요인이 될 것이다. 마찬가지로 시스템 구동에 필요한 동적 에너지 감소 역시 저전력 설계에서 주요한 부분으로 고려되어야 한다.

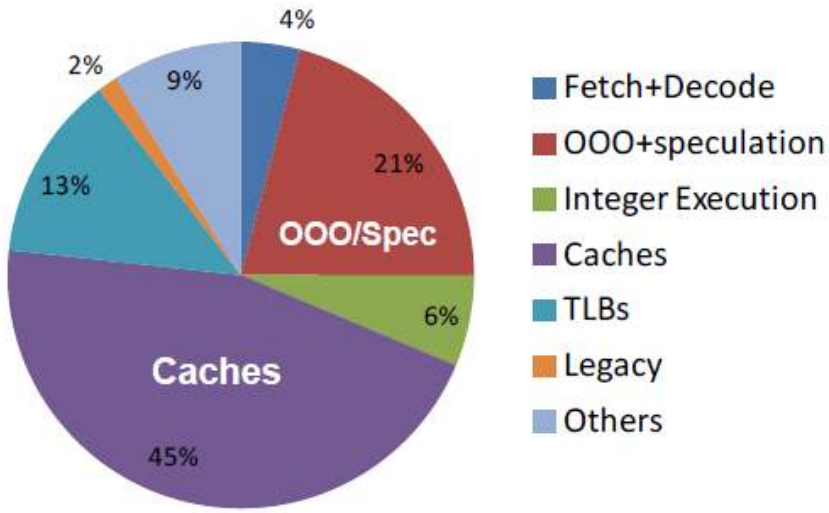

그림 1.3 Non Compute-heavy Application

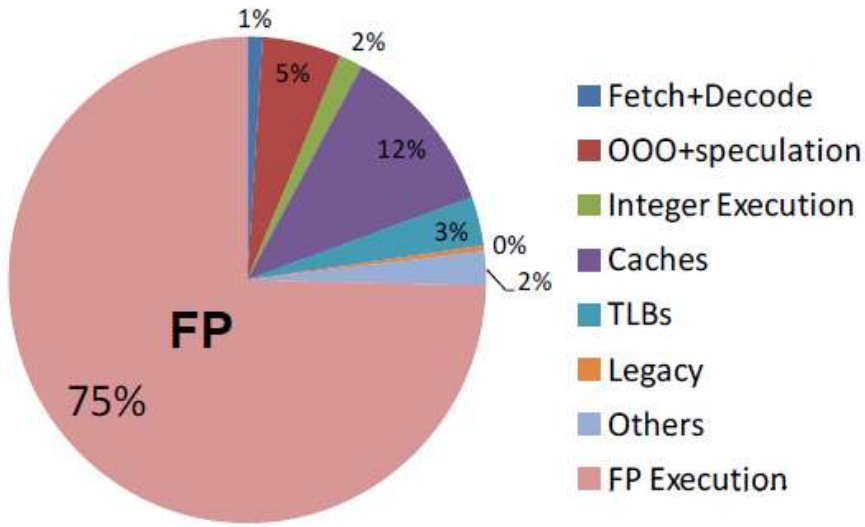

그림 1.4 Compute-heavy Application

내장형 프로세서에서는 에너지 감소를 위해 전력 관리 단계를 두어 SRAM Cell 에 공급되는 Vdd나 Vth를 점차 감소시키는 기법을 이미 수 년 전부터 사용 중에 있다. (표 1.1)

이 기법은 DVFS(Dynamic Voltage Frequency Scaling)이라 불리는데, Vdd의 감소는 전류량을 감소시키기 때문에 고속 동작에 어려움을 가져오고 이는 동작 주파수(Frequency)의 감소로 인한 속도 저하를 가져온다. 따라서 프로세서 사용량이나 배터리의 상태를 관찰하여 동작 속도가 더 느린 이른바 '대기 모드(저전력 모드)'에 돌입하게 된다. 내장형 프로세서에서는 유저의 사용 형태에 따라 프로세서 사용량의 차이가 크기 때문에 DVFS와 같은 형태의 전력 관리를 사용할 수 있으나, 수퍼 컴퓨터나 서버와 같은 고성능 프로세서를 요구하는 환경에 이를 적용하기에는 어려운 점이 있다. 이에 따라 컴퓨터의 구조 자체에 변형을 가져와 캐시 메모리의 사용 전력을 줄이고자 하는 여러 연구들이 있었다.

표 1.1 여러 프로세서들의 DVFS 정책

	Commercial			Academic	
Processors	Transmeta Crusoe (LongRun)	AMD Mobile K6 (PowerNow)	Intel PXA250	UC Berkely (ARM8)	Unicom LART (StrongARM)
Scaling Level	200~700MHz 1.1~1.65V	192~588Mhz 0.9~2.0V	100~400MHz 0.85~1.3V	5~80Mhz 1.2~3.8V	59~251Mhz 0.79~1.65V
Scaling Time	1.1↔1.65V <300μs	0.9↔2.0V 200μs	Each step 500μs	1.2↔3.8V 520μs	59↔251Mhz: 140μs 0.79→1.65V: 40μs 0.79←1.65V: 5.5ms
Scaling Power	Unknown	Unknown	Unknown	130μJ	Unknown

프로세서 레지스터와 캐시의 저장 단위 차이를 이용하여 동적 전력 사용량을 줄이고자 하는 연구로 워드 격리 캐시(Word Interleaved Cache)가 있었다. 워드 격리 캐시는 집합 연관 캐시(Set Associative Cache)에서 일반적으로 웨이당 한 블록이 저장되던 것을 블록 별로 워드의 위치에 맞추어 저장할 수 있게 하였다. 하지만 이 방법을 사용하려면 프로세서 레지스터의 크기로 캐시의 저장 단위를 나눈 몫에 해당하는 웨이 수를 가지는 집합 연관 캐시(일반적인 경우 4-웨이)로 하드웨어 제한이 있고, 읽기 성공 때에는 동적 전력의 획기적인 감소를 가져올 수 있지만 쓰기가 발생했을 때의 동작이 복잡하고 오버헤드가 크다. [KM08]

프로세서 레지스터와 L1 캐시 사이에 L0 캐시에 해당하는 추가 저장장치를 두어 L1 캐시의 동적 전력 사용량을 줄이고자 하는 연구로 필터 캐시(Filter Cache)가 있었다. 필터 캐시는 L1 캐시보다 상대적으로 크기가 작기 때문에, 필터 캐시에서 적중이 일어났을 경우 L1 캐시의 동적 전력 사용량을 줄일 수 있다. 하지만 필터 캐시는 필수 경로(Critical Path)에 위치하여 적중되지 않았을 경우 성능이 감소하며, 시간적 공간적 연관성이 상대적으로 작은 데이터 캐시의 경우, 필터 캐시 적중률이 낮기 때문에 성능 감소의 폭이 크다. [KGM97]

위상 캐시(Phased Accessed Cache)와 같이 캐시의 태그 어레이와 데이터 어레이를 분리시켜 동적 전력 사용량을 줄이고자 하는 연구도 있었다. 위상 캐시에서는 n-웨이 집합 연관 캐시에서 상대적으

로 동작 전력이 작은 태그 어레이를 먼저 동작시켜 적중여부를 알아내고, 그 후 적중이 된 웨이의 데이터 어레이만 작동시켜 적중이 되었을 시에 전통적인 캐시 모델에 비해 1/n의 데이터 어레이 동적 전력만을 사용한다. 하지만 필터 캐시와 마찬가지로 필수 경로에 위치하기 때문에 성능이 감소하며, 적중 시에 성능이 감소하기 때문에 동적 전력 소모량에 비례하여 성능 감소량이 커진다. [HKY+95]

정적 에너지를 줄이기 위한 대표적인 기법들은 주로 공급전압을 줄임으로 해서 누설 전력을 줄이는 방식을 취하고 있다. 이 기법들 중에 Vdd 단락 캐시(gated-Vdd cache)은 SRAM 내부의 그라운드 단자에 단락된 Vdd(Gated-Vdd)를 인가해 자주 사용되지 않는 캐시 라인의 내부 전압 공급을 차단하여 누설 전류를 차단하는 방식이다. 그러나 이 경우 해당 캐시 라인의 정보는 소멸되는 단점을 가진다. 부패 캐시(decay cache)에서도 일정한 시간동안 쓰이지 않는 캐시 라인을 설정해 저전력(sleep) 모드로 전이시켜 누설전력을 차단하는 기법이다. 이 기법 또한 셀 내부에 저장된 정보를 잃어버리기 때문에 재접근을 할 경우 캐시 적중률(cache hit ratio)이 떨어지고 성능저하가 발생하게 된다. [PYF+00][KHM01]

드라우지(drowsy) 기법은 두 가지 상태의 단계를 두고 있다. 먼저 캐시에 정상적인 전압을 각각 캐시 라인에 공급하는 정상전압 모드(normal mode)와 누설전력을 줄이면서 데이터의 저장 상태를 유지하기 위해 캐시 라인에 낮은 전압을 공급하는 드라우지 모드(drowsy mode)의 단계를 두고 있다. 캐시 라인이 드라우지 모드일

때 데이터를 요구할 경우 먼저 캐시 라인을 정상상태의 전압으로 승압시켜주는 깨움(wake-up)과정을 거친 후에 접근을 해야 된다. 그러나 이러한 깨움 과정에서 추가적인 사이클이 소비되고 결국 성능 저하의 단점을 내포하고 있다. [FKM+02]

이와 같이 컴퓨터 구조에 변형을 가하는 연구들의 경우에도 성능을 희생시켜야 동적·정적 전력 소모량을 줄일 수 있음을 알 수 있다. 따라서 본 연구에서는 성능의 희생을 최소화하면서 전력 소모량을 극대화 할 수 있는 기법에 관한 다음과 같은 연구를 수행한다.

1.2 연구 내용

마이크로프로세서의 트랜지스터 집적도가 높아지고 공정 단위가 작아지며 기하급수적으로 늘어난 전력 소모량으로 인해, 프로세서 캐시의 저전력 설계는 더욱 중요해 졌다. 성능 감소를 최소화 하는 한도 내에서 전력 소모량을 줄이려는 척도로 마련된 에너지-지연시간 곱(Energy-Delay Product)의 최적 값을 얻기 위해 다양한 저전력 캐시 연구가 진행되었다. 프로세서 레지스터와 캐시 저장단위가 차이난다는 점을 이용하여 캐시의 일부분만 접근하도록 고안된 워드 격리 캐시와, L0에 해당하는 위치에 작은 저장 공간을 추가한 필터 캐시, 그리고 L1의 태그 어레이와 데이터 어레이를 순차적으로 접근하도록 한 위상 캐시는 동적 전력 소모량을 줄이고자 하는 연구였고, 저전압 드라우지 모드를 추가하여 정적 전력 소모량을 줄인 드라우지 캐시 연구가 있었다. 하지만 이러한 연구들은 서로간의 장점을 서로 교환하지 못하고 각각의 방식으로 최적화되어 있었다. 따라서 이런 문제점들을 해결하기 위한 노력으로 다음과 같은 두 가지 사항을 제안한다.

첫째, 동적 전력을 감소시키는 새로운 기법을 제안한다. 선택적 워드 접근(Selective Word Reading)이라 명명한 이 기법은 상위 저장 장치에서 메모리 요청을 하였을 때, 그에 필요한 부분만을 하위 저장 장치에 접근하여 전달하여 캐시의 동적 전력 감소를 달성하였다. 전술한 워드 격리 캐시(Word Interleaved Cache)와 같이 프로세서 레지스터와 각 레벨별 캐시의 저장단위 사이에 차이가 있다는 점을

이용하지만, 하드웨어 제한을 상당히 제거하고 쓰기 접근이 일어날 때의 오버헤드를 완전히 제거하였다. 그에 더하여 관념적인 캐시의 구조뿐만 아니라 실제 캐시 구조에서의 적용 방법에 논하여 좀 더 현실성 있고 실현 가능한 기법을 제안하였다.

둘째, 정적 전력을 감소시키는 드라우지 캐시 기법을 여러 다른 기법들과 융합시켜 그들 간의 시너지 효과를 끌어낸다. 드라우지 캐시 기법, 필터 캐시 기법 그리고 위상 캐시 기법은 모두 성능 상의 손해를 보며 동적·정적 전력 감소를 이끌어낸다. 이 기법들은 병렬적으로 동시에 적용 가능하지만, 단순하게 결합만 했을 경우 성능의 손해가 매우 커져 실제 환경에 적용시킬 수 없다.

본 논문에서는 각 기법들을 적절하게 병렬화하고 구조를 변경하여 한 가지 기법을 사용 했을 때의 오버헤드만으로 세 가지 기법 모두의 장점을 극대화시킬 수 있는 방법을 제안한다. 이 방법은 첫 번째로 제안한 선택적 워드 접근 기법과 병렬적으로 사용하여 더 큰 전력 소모량의 감소를 이끌어 낼 수 있다.

1.3 논문 구성

이하 본 논문의 구성은 다음과 같다. 2장에서는 관련 연구들을 설명하며 3장에서는 선택적 워드 접근(Selective Word Reading, SWR) 기법을 제안한다. 4장에서는 워드 필터를 사용한 순차적·선택적 워드 접근 드라우지 캐시(Sequential-SWR-Drowsy Cache with Word Filter, SSDF) 기법을 제안하고 5장에서는 모의실험을 수행하고, 실험 결과를 분석한다. 끝으로 6장에서는 본 논문에 대한 결론 을 제시한다.

제2장 관련 연구

2.1 동적 전력 감소 기법

캐시 메모리의 동적 전력 소모량을 줄이기 위해 많은 연구가 진행되어왔다. 이미 많은 프로세서에서 캐시 셀의 동작 전압을 조절하는 DVFS (Danamic Voltage-Frequency Scaling)의 기법을 도입하여 사용하고 있다.

$P=V^2 \cdot f$ (P: 전력, V: 전압, f: 클락 수)

라는 식에 의해 전력소모는 전압의 제곱과 클락 수의 곱에 비례하는 형태로 나타나고, 그로 인해 전압과 클락 수를 줄이는 것에 초점을 맞추어 다수의 연구들이 진행되었다. 궁극적으로는 0V에 수렴하는 전압으로 구동되는 회로가 이상적이며 공급전압이 문턱전압보다 낮은 상태에서 동작하는 문턱아래 회로(Subthreshold Circuit)에 대한 연구도 많이 되고 있으나 낮은 전기장(E-field)으로 인해 전류량이 너무 작아 고속 동작에는 적합하지 않다. 인텔은 32nm 자사 공정에서 그림과 같이 0.34V를 에너지-속도의 최적 전압으로 발표하고 있다. V의 감소는 성능의 감소를 담보한 상태에서 진행되는 기법이기에 일명 대기모드가 아닌 일반모드에서의 전력 감소에는 한계가 있다. 그에 따라 컴퓨터 구조 자체에 변경을 가해 캐시 메모리의 동적 전력 소모량을 줄이기 위한 연구가 진행되어왔다.

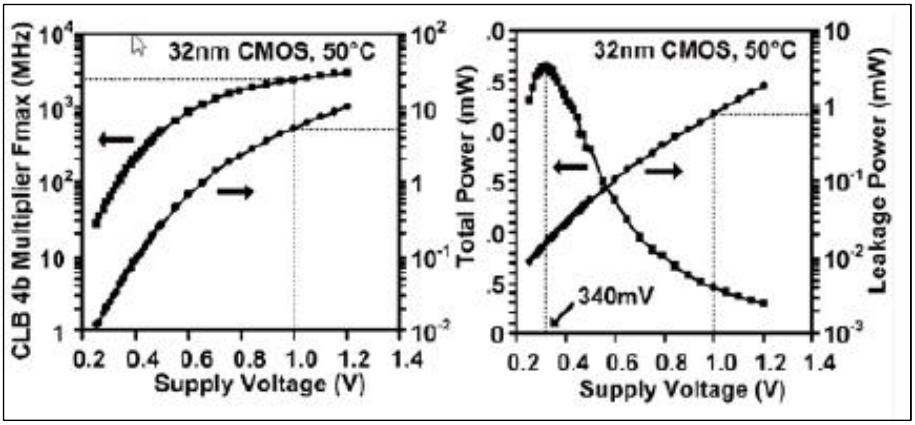

그림 2.1 인텔 32nm 공정 기술의 에너지-속도의 0.34V 최적 전압

동적 에너지를 줄이는 기법으로 대표적인 구조는 필터구조이다. 필터구조는 L0 캐시로써 L1캐시 위에 작은 캐시를 두어 소비전력을 줄이고자 하였다. 필터 캐시는 L1 캐시보다 상대적으로 크기가 작기 때문에, 필터 캐시에서 적중이 일어났을 경우 L1 캐시의 동적 전력 사용량을 줄일 수 있다. 하지만 필터 캐시는 필수 경로(Critical Path)에 위치하여 적중되지 않았을 경우 성능이 감소하며, 시간적 공간적 연관성이 상대적으로 작은 데이터 캐시의 경우, 필터 캐시 적중률이 낮기 때문에 성능 감소의 폭이 크다. [KGM97]

블록 버퍼링은 필터 캐시와 유사하나 단지 하나의 엔트리만을 가지고 있어 소비전력을 더욱 줄일 수 있는 구조이다. 일단, 한 워드가 캐시에 접근을 하게 되면 그 워드를 포함한 하나의 캐시 라인이 블록버퍼로 전송이 된다. 만약 다음 실행 때 연속으로 접근을 하게 되면 블록버퍼에서 적중이 일어나고 소비전력을 획기적으로 줄일 수 있으나 접근 실패가 일어나면 정상적인 캐시 접근으로 효과가

없다. 블록 버퍼 역시 필터 캐시와 마찬가지로 메모리 접근 필수 경로에 위치하게 되어 추가적인 싸이클이 필요하지만, 이에 관련된 논문들은 L1 캐시 추가 접근 시간에 대해 모호한 입장을 취하고 있어 논란의 여지가 있다. [GK99]

위상 캐시(Phased Access Cache)는 먼저 태그 어레이에 접근을 한 후 그 중 적중이 된 태그에 해당하는 웨이의 데이터 어레이에만 접근하는 기법이다. 전통적인 집합 연관 캐시(Set Associative Cache)에서는 태그 비교와 데이터 어레이의 접근이 동시에 일어나는데, 이 과정을 단계적으로 접근하여 동적 소비 전력을 줄이고자 하였다. 그러나 이 기법은 모든 메모리 접근에 대해 태그 비교를 위한 사이클과 적중 후 데이터에 접근하는 사이클의 순차적인 동작으로 더 많은 시간이 필요하기 때문에 전체 캐시 성능에 악영향을 미친다. [HKY+95]

순차적 예측 캐시(Predictive sequential associative cache)는 몇 가지 예측 메커니즘에 의해 찾고자 하는 블록을 선택한다. 2-웨이 집합 연관 캐시에서 작동하는 이 방식은 전통적인 방식의 2-웨이 집합 연관 캐시보다 더 높은 적중률을 보이고 같은 크기의 직접 사상 캐시(Direct-mapped Cache)보다 더 빠른 접근 시간을 보이고 있다. [CGE96]

웨이 예측 캐시(Way-predicting set-associative cache)는 순차적 예측 캐시와 유사하지만, 4웨이 이상에서 동작 가능한 예측 알고리

즘을 통해 접근하는 태그와 데이터의 웨이를 선택한다. 선택된 캐시 웨이를 먼저 동작시키고 예측이 틀렸을 경우, 다음 사이클에 다른 웨이들을 동시에 접근하는 방식을 사용하고 있다. 이러한 예측을 위하여 각각의 셋(set)마다 MRU(Most Recently Used) 알고리즘을 이용하여 해당 셋에 대한 접근시 MRU 비트를 이용하여 최근에 적중이 발생한 웨이를 먼저 접근하는 방식을 사용하고 있다. 이 연구는 예측이 성공했을 시 1사이클, 실패했을 시 2사이클의 시간이 걸리기 때문에 예측 정확도에 의해 그 성능이 제한이 된다. [IIM99]

반응 연관 캐시(Reactive-associative cache design)는 선택적 집적 사상(selective direct mapping)과 웨이 예측(way prediction)을 모두 사용한다. 데이터 캐시에서의 웨이 예측 적중률이 매우 낮기 때문에 저자는 선택적 매핑 기술을 사용하여 접속이 많이 되는 블록들을 직접 사상(direct-mapped) 위치에 저장하고, 충돌이 나는(conflicting) 블록들만 집합 연관(set-associative) 위치에 저장하였다. 이 기법들을 사용하여 해당 연구는 4-웨이 집합 연관 캐시에 대해 같은 크기의 직접 사상 캐시에 근접한 성능을 나타내었다. 하지만 이 기술은 적중 실패가 발생했을 때에도 항상 첫 번째 웨이에 접속해야하기 때문에 추가적인 접근 시간을 필요로 한다. [BV01]

선택적 캐시 웨이(Selective cache ways)에서는 응용 프로그램의 요청에 따라 캐시 웨이가 선택적으로 활성화된다. 몇몇 응용 프로그램에서는 전체 캐시의 모든 웨이가 필요하지 않을 수 있고, 그런 경우에 몇몇 웨이를 동작하지 않는 상태로 만들어 큰 성능상의 불이

익 없이 전력 소모량을 감소시킬 수 있다. 그러나 이 기법은 응용 프로그램에 대한 프로파일링(profiling)이 필요하고, 전체 웨이가 모두 활성화 되어야 하는 응용 프로그램에 대해서는 전력에서의 이득이 없다. 그리고 웨이의 수가 줄어들어 적중률이 낮아질 수 있다는 단점이 있다. [Albo99]

웨이 멈춤 캐시(Way-halting Cache)는 모든 웨이의 태그 비트의 앞부분 몇 비트를(lowest-order tag bits) 작은 완전 연관 캐시(Fully Associative Cache)에 담아둔다. 이 캐시는 셋-인덱스 디코딩을 병렬적으로 동시에 진행하며 만약 완전 연관 캐시에서 미스가 발생하게 되면 해당 캐시 웨이의 접근은 멈춘다. [ZVY+05]

프로세서 레지스터와 캐시의 저장 단위 차이를 이용하여 동적 전력 사용량을 줄이고자 하는 연구로 워드 격리 캐시(Word Interleaved Cache)가 있었다. 워드 격리 캐시는 집합 연관 캐시(Set Associative Cache)에서 일반적으로 웨이당 한 블록이 저장되던 것을 블록 별로 워드의 위치에 맞추어 저장할 수 있게 하였다. 그림 2.2에서 볼 수 있듯이 4개의 워드로 이루어진 L1 데이터 어레이의 블록들은 분할되어 4개의 웨이에 각각 저장이 된다. 프로세서가 특정 메모리 주소에 읽기 접근하려 하면 블록 오프셋의 일부를 분석하여 어떤 웨이에 접근해야 할지 결정하고 해당 웨이를 제외한 다른 웨이들을 비활성화하여 동적 전력 소모량을 줄인다. 하지만 이 방법을 사용하려면 일단 프로세서 레지스터의 크기로 캐시의 저장 단위를 나눈 몫에 해당하는 웨이 수를 가지는 집합 연관 캐시(일반

적인 경우 4-웨이)로 하드웨어 제한이 있다. 따라서 L1 캐시를 2-웨이로 설정했을 경우에 동적 전력 감소량의 이득은 4-웨이로 설정했을 경우에 비해 절반 정도가 되고, L2 캐시에 해당 기법을 적용시키려 할 때에도 L1캐시의 저장단위로 L2캐시의 저장단위를 나눈 몫을 L1 캐시의 웨이 수에 곱하여 나온 수로 L2 캐시의 웨이 수를 고정시켜야 한다. 또한 읽기 성공 때에는 동적 전력의 획기적인 감소를 가져올 수 있지만 읽기 실패가 발생했을 때, 블록을 분할하여 캐시에 배치해야하기 때문에 동작이 복잡하고 오버헤드가 크다. [KM08]

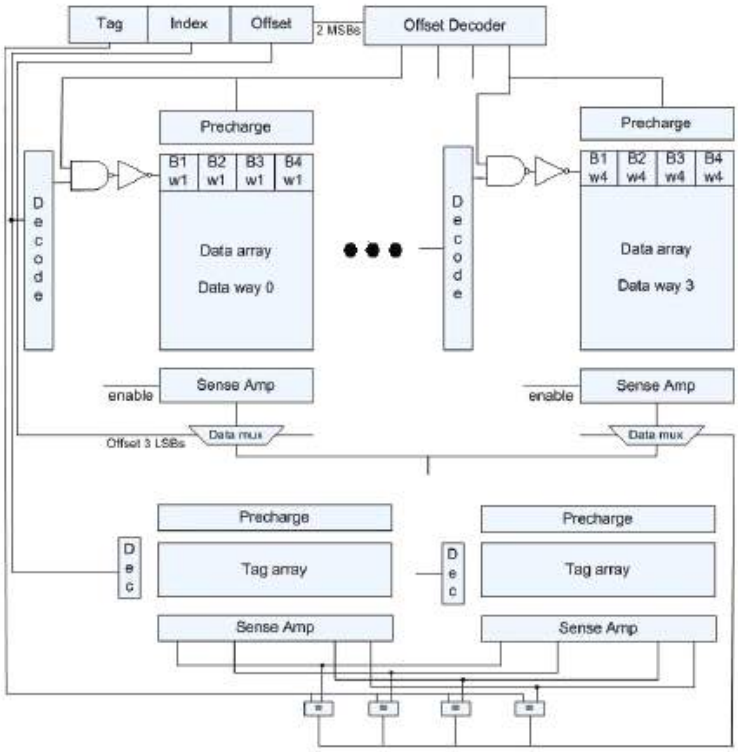

그림 2.2 워드 격리 캐시의 구조

2.2 정적 전력 감소 기법

정적 에너지를 줄이기 위한 기법은 크게 내용 저장(State Preserving) 방식과 내용 미저장(Non-State Preserving) 방식으로 나뉜다. 내용 미저장 방식은 내부 전압 공급을 차단하여 해당 캐시 라인의 정보가 소멸되는 방식이고, 내용 저장 방식은 저전력 모드에 해당하는 Vdd를 인가하여 정보를 보존하는 방식으로 일반적으로 정적 에너지 감소량은 내용 미저장 방식이 더 크나 성능 감소 또한 내용 미저장 방식이 더 크다.

2.2.1 내용 미저장 방식

2001년에 내용 미저장 캐시로 동적 크기 조절 캐시(Dynamically ResIzable Cache, DRI)가 연구되었다. 이 기법은 캐시의 적중 실패율이 일정 이상을 넘을 경우 불필요하다고 판단된 캐시 라인을 저전력 모드로 바꾸어 정적 에너지를 줄이는 기법으로 명령어 캐시(instruction cache)를 위해 고안되었다. [YPF+01]

내용 미저장 캐시의 대표적인 예인 부패 캐시(Decay Cache)는 데이터 캐시의 정적 에너지를 줄이기 위해 고안되었다. 적절하게 제안된 부패 시간 간격(decay interval) 동안 접근되지 않은 캐시 라인은 전력 단락 기법에 의해 저전력 모드로 돌입하게 된다. [KHM01]

많은 프로세서들이 성능 상의 이유로 2단계 이상의 계층적 캐시

구조를 가지고 있다. 이러한 캐시 계층을 고려한 저전력 캐시도 제안되었다. 상위 단계 캐시에서 미스가 발생하여 하위 단계 캐시에서 해당 내용을 상위 단계 캐시로 옮기게 되면, 단일 프로세서에서는 더 이상 하위 단계 캐시에서 그 내용을 가지고 있을 필요가 없다. 그렇기에 이러한 경우, 하위 단계 캐시의 해당 부분을 전력 단락 기법에 의해 저전력 모드로 바꾸면 정적 에너지의 감소를 가져올 수 있다. [LKT+02]

2.2.2 내용 저장 방식

내용 저장 방식의 대표적인 예인 드라우지(drowsy) 기법은 인가되는 전압에 따라 두 개의 캐시 모드를 가지고 있다. 먼저 캐시에 정상적인 전압을 각각 캐시 라인에 공급하는 정상전압 모드 (normal mode)와 누설전력을 줄이면서 데이터의 저장 상태를 유지하기 위해 캐시 라인에 낮은 전압을 공급하는 드라우지 모드(drowsy mode)의 단계를 두고 있다. 캐시 라인이 드라우지 모드일 때 데이터를 요구할 경우 먼저 캐시 라인을 정상상태의 전압으로 승압시켜주는 깨움(wake-up)과정을 거친 후에 접근을 해야 된다. 그러나 이러한 깨움 과정에서 추가적인 사이클이 소비되고 결국 성능 저하의 단점을 내포하고 있다. [KFB+04][FKM+02][KFB+02]

부패 캐시와 드라우지 캐시 모두에서 저전력 모드에 돌입하는 방법으로 많이 사용하는 정책은 단순 정책(Simple Policy)과 비접속 정책(No-access Policy)이다. 단순 정책은 특정 시간 간격마다 모

든 라인을 저전력 모드로 바꾸고 메모리 접근이 일어나는 라인에 대해 일반 모드로 전환하는 방식이고, 비접속 정책은 특정 시간 간격동안 접근이 일어나지 않은 라인을 저전력 모드로 바꾸는 방식이다. 이러한 정책적인 부분을 집합 연관 캐시에서 셋 별로 가장 최근에 접근한 라인은 저전력 모드로 바꾸지 않는 방식 등 여러 가지 방안으로 수정한 연구들이 계속되고 있다. [PSS+05][ZZT+08]

이 외에도 2011년에 웨이-라인 예측 유닛을 이용한 드라우지 캐시가 제안되었다. 이 기법은 PC(Program Counter)값을 기반으로 예측 테이블을 유지하고, 파이프라인에서 주소 계산 전 단계에 웨이-라인 예측과 깨움 과정을 거침으로 메모리에 접근하는 시점에서는 예측의 성공 여부와 관계없이 모든 블록이 정상 전압 모드에 들어가 있게 된다. 따라서 이 기법을 유지하면 예측의 성공률에 의해 정적 에너지 소모량이 차이가 나지만 성능의 하락은 경험하지 않아도 된다. [Lee11]

그림 2.3 은 드라우지 캐시 라인의 구성 모습을 보여주고 있다. 각 웨이의 셋 별로 드라우지 비트를 한 개씩 가지고, 각각의 셋이 어떤 모드에 있는지를 판별한다. 전압 조종 장치 부분에서는 1V의 정상 전압과 0.3V의 저전력 전압을 선택하여 SRAM에 공급한다. 저전력 전압을 공급할 경우 해당 셋은 정상 전압 모드에 비해 10% 정도의 정적 에너지만을 소모한다. 대신 저전력 모드에 있는 셋에 접근할 때 깨움(wake-up) 과정을 위해 1 싸이클의 시간과 추가적인 에너지가 필요하다.

이 기법은 2002년에 저전력 내장형 프로세서의 최대 생산회사인 ARM에서 발표하였다. 하지만, 당시의 기술로는 0.3V 정도에서 에러율을 최소화하면서 SRAM을 구동할 수 없었고, DVFS 기술의 사용 시에도 0.7V 정도의 최저 전압을 구동하고 있었기 때문에 현실성이 부족했다. 2011년에 인텔은 32nm 자사 공정에서 0.34V를 에너지-속도의 최적 전압으로 발표했고, 퀄컴(Qualcomm)에서 45nm 공정을 사용, 0.54V까지 동작하는 SRAM을 개발하여 사용 중에 있다. 이와 같이 저전압 트랜지스터 기술의 발전으로 드라우지 캐시의 실현 가능성도 한층 높아졌다고 본다. [Park11]

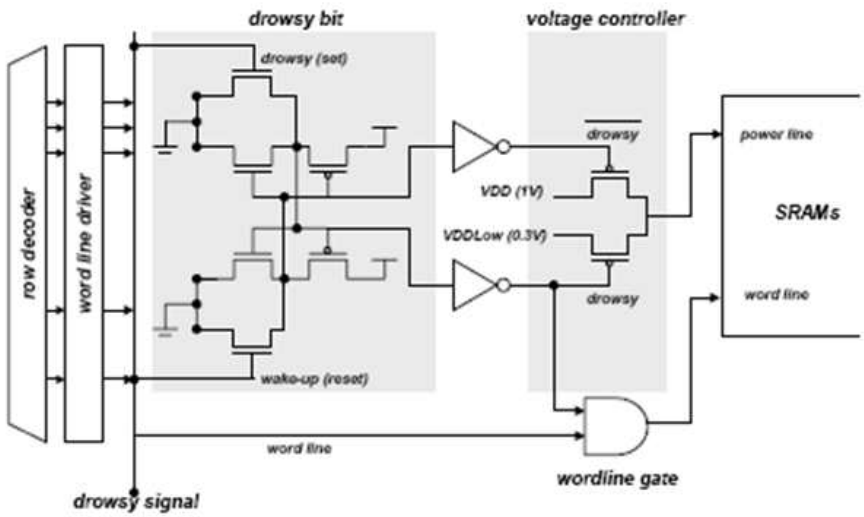

그림 2.3 드라우지 캐시 라인의 구현 모습

제 3장 선택적 워드 접근 기법

3.1 개발 동기

선택적 워드 접근 기법(Selective Word Reading)은 각 저장장치의 단계에 따라 저장되는 단위가 다르다는 점에 착안했다. 캐시에서 저장되는 단위가 커지는 것은

1) 캐시의 공간적 연관성 증가로 인한 성능 향상
2) 디코더 크기의 감소와 태그 크기의 감소로 인한 전력 감소
3) 2와 같은 이유로 인한 면적 감소

와 같은 이득과 함께 또한 캐시에 저장되는 블록의 다양성을 줄어들게 하여 성능의 감소 또한 유발할 수 있기 때문에, 저장장치의 크기에 맞는 적당한 저장 단위를 선택해야 한다. 본 연구에서는 CPU Register에서 1 워드 단위(8B), 32kB L1 캐시에서 4 워드 1 블록 단위(32B), 그리고 1MB L2 캐시에서는 8워드 1블록 단위(64B)로 저장이 되고 있음을 가정했다.

표 3.1에서 1MB와 2MB 캐시 모두에서 블록 사이즈를 32B에서 64B로 증가시킬 경우 동적 에너지에서 약 4%, 정적 전력에서 약 5.5% 그리고 면적에서 약 4.7% 가량의 이득을 볼 수 있음을 확인할 수 있다.

표 3.1 캐시 블록 사이즈에 따른 이득 비교

캐시 크기와 웨이 수	1Mb 8way			1Mb 16way			2Mb 8way			2Mb 16way		
블록의 크기	32B	64B	비율 (%)	32B	64B	비율 (%)	32B	64B	비율 (%)	32B	64B	비율 (%)
접근 당 소모되는 동적 읽기 에너지 (nJ)	0.516	0.496	96.1	0.527	0.505	95.9	1.049	1.007	96.0	1.056	1.016	96.2
뱅크 한 개당 정적 전력 (mW)	588.5	554.7	94.3	590.4	556.9	94.3	1171	1110	94.8	1174	1112	94.7
총 면적 (mm^2)	3.952	3.760	95.2	3.969	3.777	95.2	8.224	7.850	95.4	8.258	7.883	95.5

서로 다른 단계에서 원하는 용량이 다르기 때문에 프로세서 레지스터에서 L1 캐시에 메모리 요청을 보냈을 때 L1 캐시에서는 4워드가 모두 활성화되어 읽히고, 그 중 필요한 워드만이 캐시 조종 장치에 있는 선택기(mux)를 통해 전달된다. L1 캐시에서 L2 캐시로 메모리 요청을 보냈을 때도 마찬가지로 L2 캐시에서는 8워드가 활성화되어 읽히지만 그 중 4워드만이 L1 캐시로 전달된다. (그림 3.1)

선택적 워드 접근 기법을 사용하면 프로세서 레지스터에서 L1 캐시에 메모리 요청을 보냈을 때 4워드를 통째로 읽지 않고 필요한 워드만 선택적으로 읽을 수 있게 되고, L1에서 L2로 메모리 요청을 보냈을 때에도 마찬가지로 8워드를 통째로 읽지 않고 필요한 4워드만 선택적으로 읽을 수 있다. 이를 통해 데이터 어레이에서 발생하는 동적 전력 소모량을 L1 캐시에서 약 25%로, L2 캐시에서 약 50%로 감소시킬 수 있다.

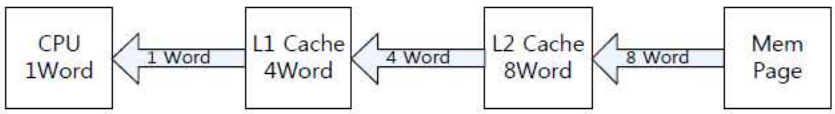

그림 3.1 각 저장 단계별 저장되는 단위와 요청 단위

3.2 구현

3.2.1 개념적 구현

그림 3.2는 전통적 캐시의 데이터 어레이이다. L1 캐시에서는 4워드 1블록 단위로, L2에서는 8워드 1블록 단위로 정보 유입(Data-in)과 정보 유출(Data-out)이 일어나고 있다. 프로세서에서는 1워드 단위로 읽기 접근(Load-word Instruction)가 발생하는데, L1에서 발생하는 4워드 정보 유출(Data-out) 중에 1워드를 선택하는 일은 L1 캐시 조종 장치에서 담당하고 있다. 마찬가지로 L1에서 L2로의 요청은 4워드 단위로 발생하는데, L2의 8워드 정보 유출(Data-out)에서 4워드를 선택하는 일은 L2 캐시 조종 장치에서 담당하고 있다. 우리가 제안하는 선택적 워드 접근 캐시의 모습은 그림 3.3 으로 캐시 조종 장치에서 행하던 워드 선택을 캐시 접속 시에 해서 필요한 부분 외의 다른 부분들을 비활성화해 동적 전력 소모를 줄인다.

상대적으로 크기가 작아 동적 전력 소모량이 적은 태그 어레이에의 접근은 전통적 캐시와 선택적 워드 접근 캐시에서 동일하게 이루어지고, 데이터 어레이에의 접근은 블록 오프셋의 상위 비트를 사용하여 그림 3.3 의 회색 부분을 비활성화 한다.

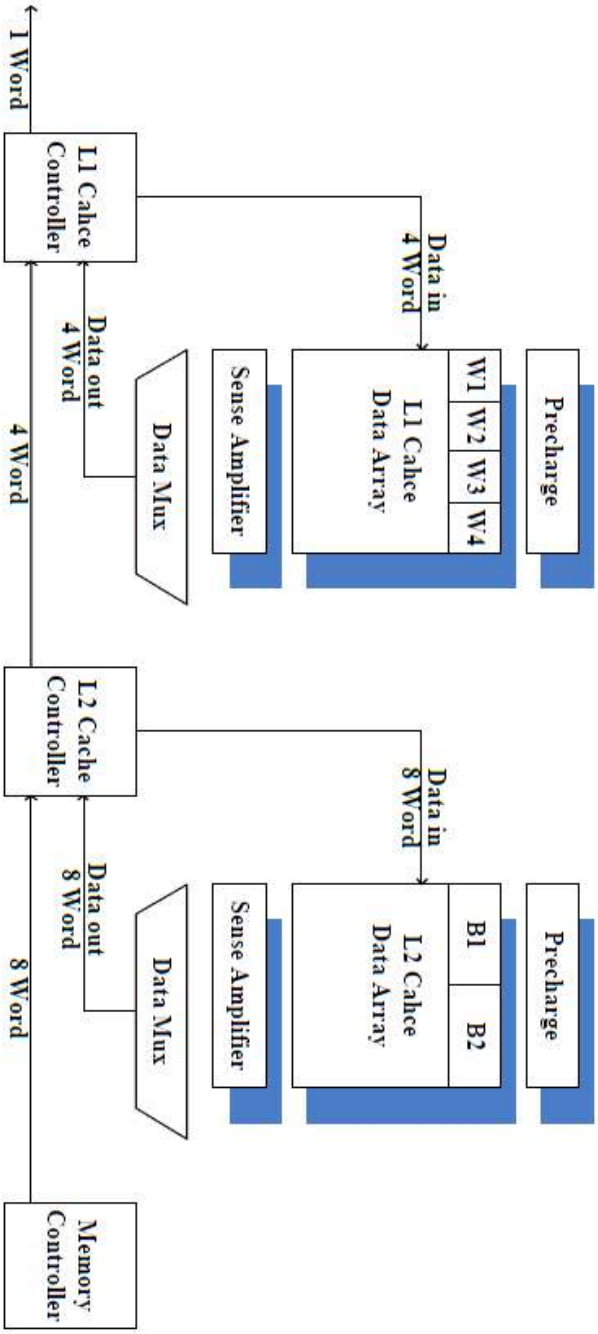

그림 3.2 Conventional 캐시의 데이터 어레이

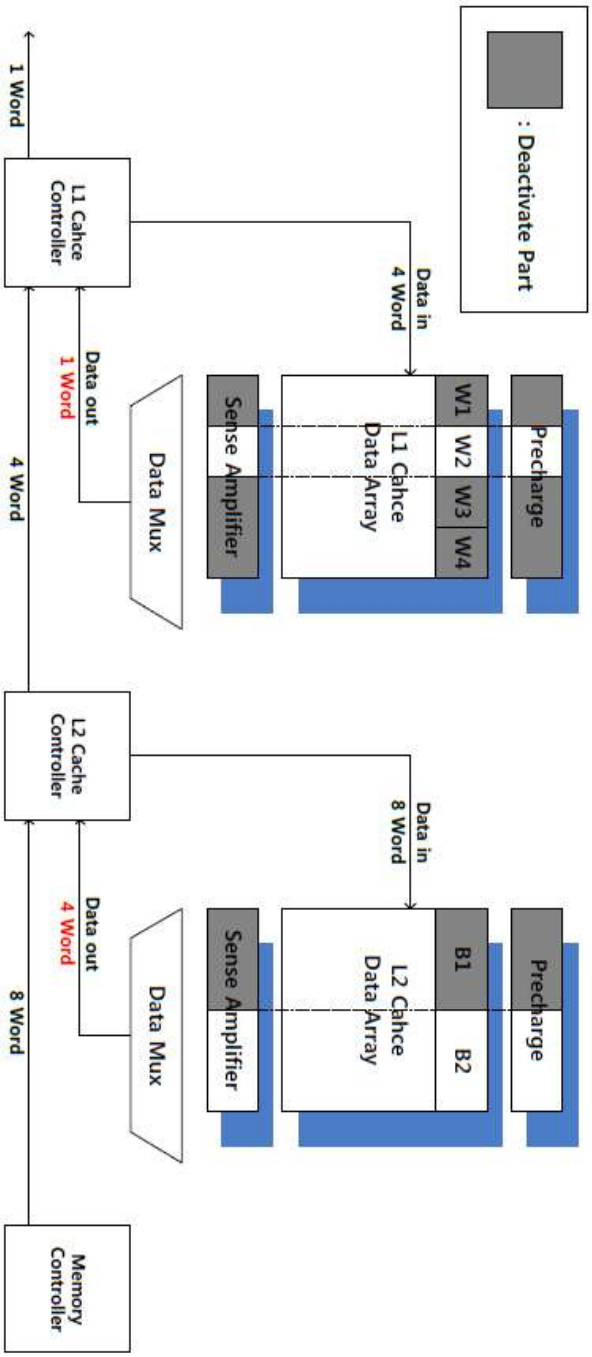

그림 3.3 SWR 캐시의 데이터 어레이

그림을 비교해보면 하드웨어적으로 차이가 있는 부분은 정보 유출 부분에 국한되는 것을 살펴볼 수 있다. 전력 소모의 이득이 발생하는 때가 이 정보 유출 버스가 사용될 때인데 이 부분은 캐시에서 읽기 접근 명령어(Load Instruction) 적중이 되었을 경우 사용된다. 읽기 적중이 발생하면 그림에서 회색 부분을 비활성화 하는데 L1 캐시의 경우 약 25%, L2 캐시의 경우 약 50% 가량의 동적 전력만을 소비하게 된다. 일반적으로 캐시의 적중률은 90%를 넘고 전체 메모리 접속 요청 중 읽기 접근 명령어의 비율 또한 90%를 넘기 때문에 선택적 워드 접근 캐시는 동적 에너지 감소에서 큰 효율을 보인다. 회색 부분을 비활성화하기 위해서는 주소의 일부분을 캐시 접속 시에 메모리 요청에 실어 보내면 되는데, L1 캐시의 경우 블록 오프셋의 상위 2bit, L2 캐시의 경우 블록 오프셋의 상위 1bit가 필요하다. 캐시에서 적중 실패가 발생한 경우 정보 유입 버스를 사용하게 되는데 이 때 전통적 캐시와 동일한 동작을 하게 되며, 워드 격리 캐시와 달리 어떠한 추가비용(Overhead)도 발생하지 않는다.

3.2.2 실제적 구현

실제 캐시의 데이터 어레이의 모습은 개념적인 모형과는 큰 차이가 있다. 최상위 단계에서 데이터 어레이는 몇 개의 독립적인 뱅크로 이루어진다. 각각의 뱅크는 동시에 접속 가능하고 각자 자신의 주소 버스와 데이터 버스를 가지고 있다. 각각의 뱅크는 다시 몇 개의 독립적인 서브-뱅크로 이루어지고 이 서브-뱅크 한 개가 캐시 접속 시에 활성화된다. 각각의 서브-뱅크는 블록의 일부분을 저장하고 있는 몇 개의 매트로 구성이 되고 한 서브-뱅크 내에 위치한 모든 매트는 접속 시에 동시에 활성화가 된다. (그림 3.4)

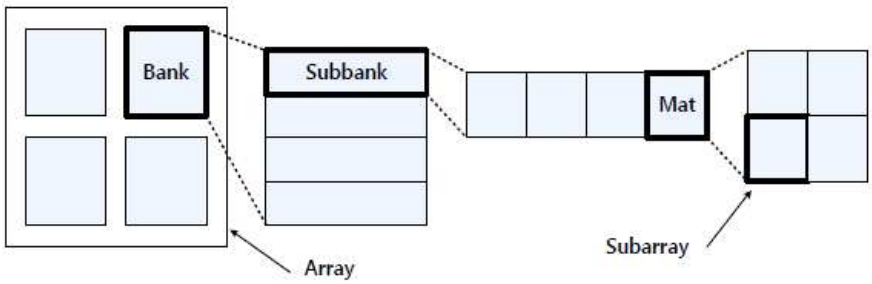

그림 3.4 4개의 뱅크를 가진 예제 데이터 어레이

매트 한 개에 저장되는 크기를 p라 하면

p = *(캐시에 저장되는 단위 / 서브-뱅크 당 매트의 개수)*

로 결정된다. 예를 들어 L1 캐시에 저장되는 단위가 32B 블록이므로, 서브-뱅크 하나 당 매트의 개수를 4개로 가정하면 한 매트 당 8B, 1워드가 저장된다.

전통적 캐시의 경우 한 번의 접근에 서브-뱅크에 있는 4개의 매트를 모두 활성화 시키고 1개의 매트 당 1워드 씩 총 합하여 4워드를 캐시 조종 장치로 보내지만, 선택적 워드 접근 캐시에서는 1개의 매트만을 활성화시켜 1워드만을 캐시 조종 장치로 보낸다. (그림 3.5)

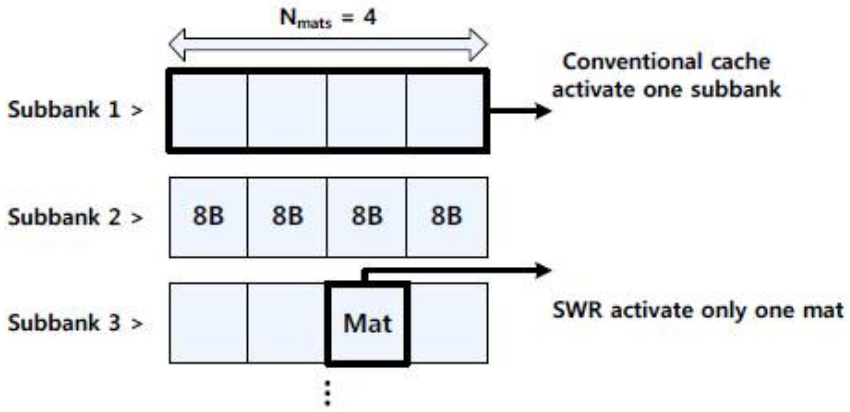

그림 3.5 데이터 어레이에서 활성화 되는 부분

$$상위\ 레벨\ 요구\ 단위 = n * p\ (n=정수)$$

를 만족할 경우 선택적 워드 접근 캐시의 에너지 효율은

$$에너지\ 효율 = 상위\ 레벨\ 요구\ 단위 / 하위레벨\ 저장\ 단위$$

로 최대가 되지만,

$$상위\ 레벨\ 요구\ 단위 < p$$

일 경우 선택적 워드 접근 기법의 효율은

$$에너지\ 효율 = 1 / 서브-뱅크\ 당\ 매트\ 개수$$

가 된다.

따라서 레지스터에서 8B, L1 캐시에서 32B 그리고 L2 캐시에서 64B를 요구 할 때 에너지 효율을 최대로 만들기 위해 L1 캐시의 서브-뱅크 당 매트 개수는 4개 이상, L2 캐시의 서브-뱅크 당 매트 개수는 2개 이상이 되어야 한다. 용량이 상대적으로 큰 L2 캐시의 경우 CACTI를 통해 살펴본 최적의 서브-뱅크 당 매트 개수는 16개이므로 큰 문제가 없지만 L1 캐시의 서브-뱅크 당 매트 개수는 2개일 때 최적 값이 나오기 때문에 서브-뱅크 당 매트 개수를 4개 이상으로 지정해야만 하는 선택적 워드 접근 캐시에서는 다소간의 손해를 본다. 하지만 서브-뱅크 당 매트 개수가 4개가 되는 대신 동적 전력 소모는 1/4 가량으로 떨어지기 때문에 선택적 워드 접근 캐시에서는 더 큰 에너지 절감 효과를 볼 수 있다.

L1 기준 캐시의 경우 뱅크의 개수를 1개, 서브-뱅크의 개수를 4개, 서브-뱅크 당 매트의 개수를 4개, 전체 크기를 32kB, 블록 크기를 32B, 직접 사상 캐시라고 가정하면 한 개의 32B 블록을 찾는데 10비트의 부분 주소가 필요하다. 서브-뱅크 한 개에 매트가 4개 있으므로 매트 하나에는 8B 1워드씩 저장되어 접속 시에 서브-뱅크 내의 매트 4개가 활성화되어 8B*4 = 32B 가 정보 유출 버스로 출력된다. 따라서 서브-뱅크 4개 중 한 개를 고르면 되기 때문에 2비트가 제해져서 최종 단계에서는 8비트의 부분 주소만 있으면 된다. 캐시에서는 H-tree 알고리즘에 의해 해당 블록을 찾는데 그림에서와 같이 H0 노드로 10비트의 주소가 요청된다. H0 노드는 H1 노드로, H1 노드는 H2 노드로, 그리고 H2 노드는 V0 노드로 10비트의 주소를 요청한다. V0 노드는 위와 아래 방향 중 단방향을 정하여 9비트의 주소를 요청하고 V1 노드도 위, 아래 중 한 방향을 정하여 8비트의 주소를 매트로 요청하게 된다.

이와 같이 V노드들은 단방향을 고르고, H노드들은 단순한 중계 기능만을 하고 있어, 한 번 접속이 발생하게 되면 하나의 서브-뱅크 내의 모든 매트가 활성화된다. 예를 들어, 그림 3.6 에서와 같이 접근 대상 워드(Target Word)가 4x4 행렬 구조에서 (1,1) 위치에 있는 경우 전통적 캐시에서는 적색 박스 내에 있는 1행의 4개의 매트가 활성화된다. 선택적 워드 접근 캐시에서는 H1 노드와 H2 노드에도 방향을 고르는 기법을 추가하여 청색 라인을 따라 메모리 요청이 전송되어 (1,1) 매트만이 활성화되게 한다.

선택적 워드 접근 캐시에서는 같은 L1 기준 캐시에 대해 8B 워드

를 찾아내야 하므로 총 12비트의 부분 주소가 필요하다. H0->H1로는 12비트, H1->H2로 한 방향을 정하고 11비트(예시의 경우 좌측), H2->V0로 한 방향을 정하고 10비트(예시의 경우 우측), V0 이후는 전통적 캐시와 같다. (그림 3.6)

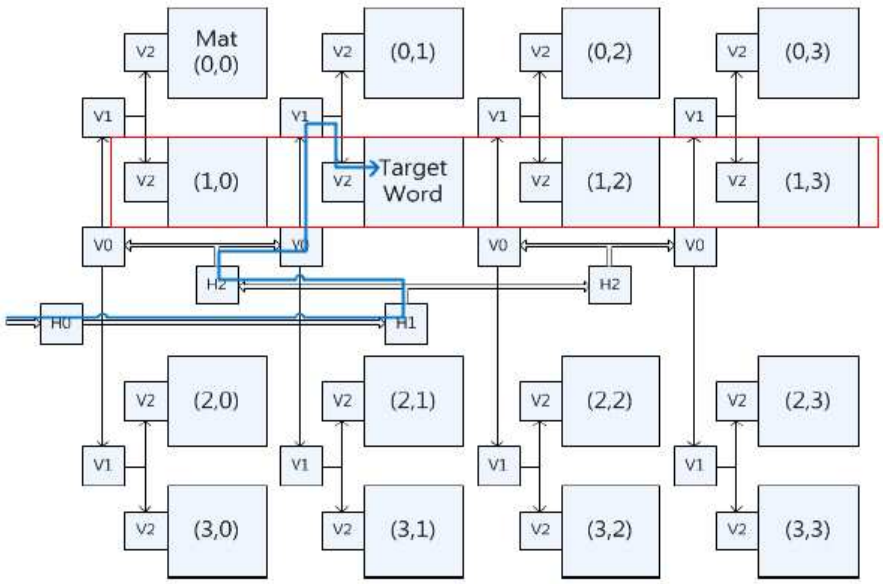

그림 3.6 뱅크에 위치한 H-tree의 구조. 박스는 conventional 캐시에서 활성화되는 부분, 라인은 SWR에서 활성화되는 부분.

3.3 전력 소모량 계산

3.3.1 전력 소모량 수식

선택적 워드 접근 캐시를 사용하면 동적 읽기 에너지(Dynamic Read Power)를 줄일 수 있다. 동적 에너지와 정적 에너지를 모두 고려한 한 저장 단계에서의 전체 소모 에너지의 수식은 다음과 같다.

$$E_{tot} = E_{dyn-read} * n(read\ hit) + E_{dyn-write} * n(write\ hit) + E_{miss} * n(miss) + P_{leak} * total\ time$$

$E_{dyn-read}$는 읽기 접근 당 동적 에너지 소모량(Dynamic Energy Per Read Access)을 의미하는데 이것은 매트에서의 에너지 소모량과 요청과 답신 네트워크(Request And Reply Network)에서의 에너지 소모량의 합으로 나타낼 수 있다. 이를 수식으로 나타내면 다음과 같다.

$$E_{dyn-read} = E_{req-net} + E_{read-mats} + E_{rep-net}$$

이 중 매트에서의 에너지 소모량은 3.2.2 절에서 예시된 L1 캐시

와 같이 뱅크의 수가 1개일 경우,

$$E_{mats} = E_{mat} * N_{mats-in-subbank}$$

와 같이 나타나는데 SWR에서 이는 (상위 레벨 요구 단위 / 하위 레벨 저장 단위)만큼 줄어들게 된다. 3.2.2 절에서 예시로 든 L1 캐시의 경우 $N_{mat-in-subbank}$ 값이 4에서 1로 줄어들게 되어 SWR 캐시의 E_{mats}는 전통 캐시의 값의 1/4로 줄어든다.

$E_{read-request-network}$는 회신 주소(request address)가 수평-수직 트리 구조 네트워크(H-V tree network)를 통하여 전달되는 동안의 에너지 소모를 의미하는데, 그 식은 다음과 같다.

$$E_{req-net} = E_{arr-to-bank-req} + E_{hor-req} + E_{ver-req}$$

여기에서 가정하는 네트워크는 3.2.2 절에서 예시된 그림 3.6에서 표현된 네트워크와 같은 모양이다. 32kB, 직접 사상, 32B 블록을 가진 L1 캐시를 가정하면 수평-수직 트리 구조 네트워크에 접근할 때 10 비트가 필요하기 때문에, 어레이에서 뱅크까지의 요청 에너지는

$$E_{arr-to-bank-req} = (10)E_{arr-to-bank-1-bit}$$

와 같이 로 표현되는데, 제안된 SWR 캐시에서는 2 비트가 추가되

어 해당 값은 (12)$E_{arr-to-bank-1-bit}$ 이 된다. $E_{hor-req}$와 $E_{ver-req}$는,

$$E_{hor-req} = E_{H0-H1} + E_{H1-H2} + E_{H2-V0}$$
$$E_{ver-req} = E_{V0-V1} + E_{V1-V2}$$

와 같이 표현된다. 그리고 전통적 기준 캐시의 경우

$$E_{H0-H1} = (10)E_{H0-H1-1-bit}$$
$$E_{H1-H2} = (20)E_{H1-H2-1-bit}$$
$$E_{H2-V0} = (40)E_{H2-V0-1-bit}$$
$$E_{V0-V1} = (36)E_{V0-V1-1-bit}$$
$$E_{V1-V2} = (32)E_{V1-V2-1-bit}$$

$$\therefore E_{hor-req} = (10)E_{H0-h1-1-bit} + (20)E_{H1-H2-1bit} + (40)E_{H2-V0-1bit}$$
$$E_{ver-req} = (36)E_{V0-V1-1-bit} + (32)E_{V1-V2-1bit}$$

와 같이 나타나고, SWR의 경우

$$E_{H0-H1} = (12)E_{H0-H1-1-bit}$$
$$E_{H1-H2} = (11)E_{H1-H2-1-bit}$$
$$E_{H2-V0} = (10)E_{H2-V0-1-bit}$$
$$E_{V0-V1} = (9)E_{V0-V1-1-bit}$$
$$E_{V1-V2} = (8)E_{V1-V2-1-bit}$$

과 같이 나타난다. 수평축 네트워크(Horizontal network)와 수직축 네트워크(Vertical network)에서 각 노드 간의 1 비트 전송에 드는 에너지는 큰 차이가 없다고 가정하고 전송되는 비트의 수로 대략적인 비교를 해보면 SWR 캐시에서는

$$E_{SWR-hor-req} \fallingdotseq (12+11+10)/(10+20+40)\ E_{hor-req}$$
$$E_{SWR-ver-req} \fallingdotseq (9+8)/(36+32)\ E_{ver-req}$$

와 같이 에너지가 소모되는 것을 알 수 있다.

또한 답신(Reply)으로 4 워드가 전송되는 것을 1 워드만 전송하게 하므로 $E_{dyn-reply-network}$ 도 1/4 로 줄어드는 것을 알 수 있다. 이 식을 종합하여 보면,

$$E_{swr-dyn-read} = E_{SWR-req-net} + (0.25)E_{mats} + (0.25)E_{rep-net}$$
$$E_{SWR-req-net} = (0.83)E_{arr-to-bank} + (0.47)E_{hor-req} + (0.25)E_{ver-req}$$

으로 총 L1 캐시에서의 동적 에너지 소모량은 약 32.48%로 감소한다. 마찬가지의 방법으로 L2 캐시에서 계산해보면 동적 에너지 소모량은 약 51.5%로 감소한다.

3.4 워드 버퍼의 고려

2.1 절에서 살펴본 바와 같이 블록 버퍼링과 같은 기법을 적용하는 것은 동적 에너지 감소에 큰 도움이 된다. SWR 캐시에 해당 기법을 적용할 때에는 버퍼의 크기가 워드 단위로 달라져야 하는데, 그 경우 적중률에 차이를 보이게 된다. 명령어 캐시에서의 워드 단위 접근과 데이터 캐시에서의 워드 단위 접근은 서로 다른 결과를 유발하게 된다. 그림 3.7은 명령어 캐시에서 블록 단위 버퍼와 워드 단위 버퍼를 설치할 경우 해당 버퍼의 적중률을 비교한 그래프이다. 블록 버퍼의 경우 82%, 워드 버퍼의 경우 47% 정도의 적중률을 가지고 있는데, 워드 버퍼의 크기가 블록 버퍼의 25% 정도밖에 되지 않음에도 불구하고 공간적 연관성이 큰 명령어 캐시에서는 블록 버퍼의 50% 이상의 적중률을 가지는 것을 볼 수 있다. 이로 인한 동적 에너지 소모량의 변화는 5장에서 분석하도록 한다.

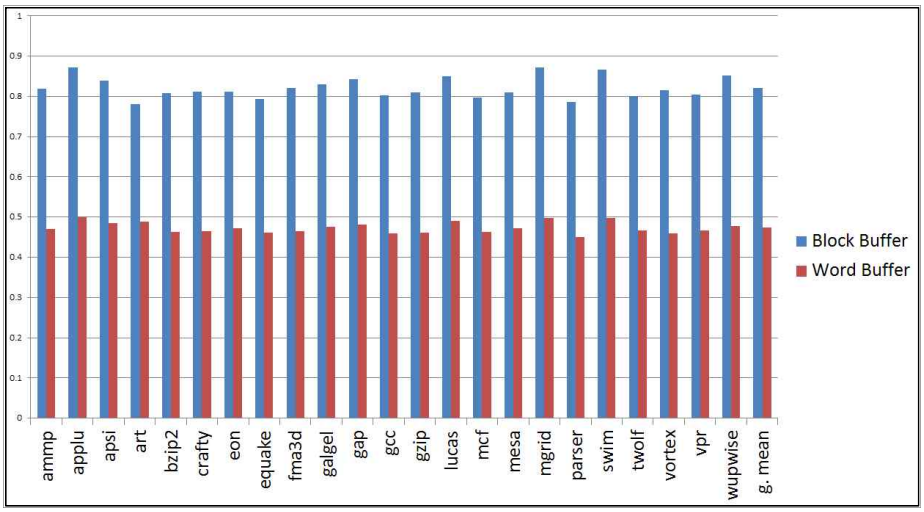

그림 3.7 명령어 캐시에서 블록 버퍼와 워드 버퍼의 적중률

RISC(Reduced Instruction Set Compute) 기계에 32bit 명령어 하드웨어를 가정하면, 1 워드에 해당하는 8 B는 64 bit이기 때문에 2개의 명령어를, 1 블록에 해당하는 32 B는 총 8개의 명령어를 한 번에 읽어올 수 있는 형태를 가지고 있다. 명령어 캐시에서 SWR 캐시를 사용할 경우 한 번에 읽어내는 명령어의 개수에 제한을 받게 되어 그에 따라 명령어 인출 너비의 최댓값을 줄여야 효율이 좋아진다. 1 워드를 읽을 경우 명령어 인출 너비는 2가 되어야 하고, 2 워드를 읽을 경우 명령어 인출 너비는 4가 되어야 한다.

SPEC 2000에서의 IPB(Instruction Per Branch)를 조사해보면 표 3.2 와 같이 나타남을 알 수 있다. 블록 상의 분기 명령어(Branch Instruction)가 존재하는 위치나 블록 상의 명령어 인출 시작 지점에 의해 IPB 값의 의미는 달라질 수 있지만, 4개의 응용 프로그램을 제외하고 IPB 값은 모두 8을 넘는 것을 볼 수 있고, 이것은 명령어 캐시에 SWR 기법을 적용시킬 경우 성능의 저하를 가져올 수 있다는 것을 이야기한다. 그림 3.8 은 명령어 인출 너비에 따른 IPC(Instruction Per Clock) 값을 조사한 그래프로, 명령어 인출 너비가 2인 경우, 8인 경우에 비해 75%로 IPC가 감소하였고, 명령어 인출 너비가 4인 경우, 8인 경우에 비해 98%의 IPC 값을 보였다.

표 3.2 SPEC 2000에서 각 벤치마크들의 IPB

	IPB (Instruction per branch)
mgrid	243.1791
swim	80.6906
applu	69.2163
lucas	61.2743
apsi	25.1076
galgel	17.7062
art	11.4866
mesa	11.4279
gzip	9.6885
twolf	9.0372
v.p.r.	9.0195
equake	8.7619
crafty	8.3895
gap	7.1280
parser	6.0826
vortex	6.0197
mcf	5.5865
a. mean1	34.6942
a. mean2 [1)]	22.7358

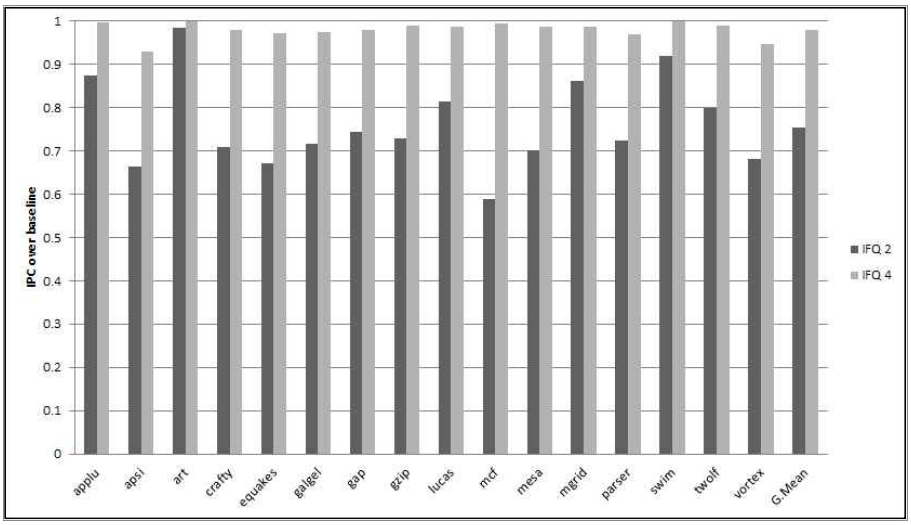

그림 3.8 명령어 인출 너비 변화에 따른 IPC

1) 최댓값과 최솟값을 제외하고 계산한 산술평균 값

데이터 캐시에서 SWR 캐시를 사용하고, 블록 버퍼가 있다고 가정하면 명령어 캐시에서와 유사한 성능 저하 유발 요인이 생긴다. SWR 캐시에서는 한번에 1 워드만을 접근하므로 블록 단위 버퍼가 아닌 워드 단위 버퍼를 유지하여야 한다. SPEC 2000에서 기준 L1 캐시를 대상으로 조사한 바에 의하면 블록 버퍼에의 적중률은 평균적으로 약 26%, 워드 버퍼에의 적중률은 평균적으로 약 10%로 관측되었다.

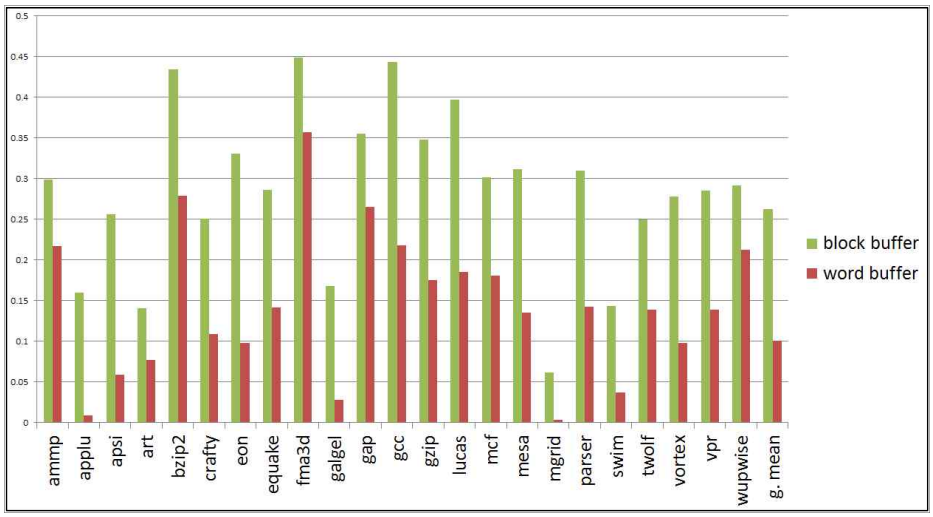

그림 3.9 데이터 캐시에서 블록 버퍼와 워드 버퍼의 적중률

제 4장 워드 필터를 사용한 순차적·선택적 워드 접근 드라우지 캐시

4.1 개발 동기

3장에서 제안된 선택적 워드 접근 캐시는 캐시의 동적 에너지 소모를 줄이는 효율적인 기법이지만, 정적 에너지 감소에는 도움이 되지 못한다.

정적 에너지 감소를 위한 알고리즘으로 2장에서 논한 여러 방법들이 있지만, 본 논문에서는 그 중 내용저장 방식의 대표적인 예인 드라우지 캐시 기법에 초점을 맞추었다. 논문의 목표가 여러 저전력 기법들의 조합 과정에서 발생하는 성능 손해를 최소화이기 때문에 성능 상에서 큰 손해를 보는 내용 미저장 방식은 적합하지 않다고 판단하였다. 드라우지 캐시 기법을 사용할 경우 저전력 모드에 있는 블록에 저장된 내용은 사라지지 않지만 해당 블록에 접근할 때 깨움 과정을 거쳐야 한다. 이 때 캐시 접근 시간에서 1 싸이클의 손해를 보게 되는데, 저전력 모드에 있는 블록 수가 늘어날수록 정적 에

너지 감소가 커지지만 전체 프로그램 수행 시간이 길어질 확률 또한 증가하게 된다.

필터 캐시와 드라우지 캐시 기법을 조합한 경우, 필터 캐시에 의해 L1 캐시의 접근이 제한되어 드라우지 상태에 놓이는 블록의 수가 많아진다. 이로 인해 정적 에너지가 추가적으로 감소하는 효과를 볼 수 있고, 상대적으로 작고 빠른 필터 캐시에 접근할 때 싸이클의 이득을 보는 효과도 얻을 수 있다. [KFB+02] 하지만, 필터 캐시 자체가 필수 경로를 지나치기 때문에 필터 캐시에서 적중 실패하고 저전력 모드에 있는 블록에 접근할 경우 드라우지 캐시만 구현했을 때에 비해 1 싸이클의 추가적인 접근 시간 손해를 본다. 따라서 필터 캐시의 적중률이 낮은 데이터 캐시에 이 기술을 적용하기는 힘들다.

순차적(위상) 캐시를 구현할 경우, 태그 어레이에 먼저 접근한 이후 적중이 발생했을 경우에만 데이터 어레이에 접속하면 되기 때문에, n-웨이 연관 집합 캐시를 가정 했을 경우 데이터 어레이를 접속할 시 전통적 캐시 모델에 비해 1/n 에 해당하는 데이터 어레이 접속 동적 에너지만 있으면 된다. 하지만, 필터 캐시나 드라우지 캐시에서와 마찬가지로 캐시 적중 시에 접근 시간에서 1 싸이클의 손해를 보게 된다. 이 때문에 과거에는 8 이상의 웨이 수를 가진 L2 캐시에서 동적 에너지 감소율이 높기 때문에 많이 사용되어왔지만, L1 캐시의 웨이 수가 4이상을 가지게 되는 최근 추세에서 이 기법의 L1 캐시에의 적용 방법에 대해 논해볼 필요가 있다.

순차적 캐시와 드라우지 캐시 기법을 조합한 경우, 태그 어레이에서 실제 접근이 적중했을 경우에만 저전력 모드에서 정상전력 모드로 깨울 수 있기 때문에 필터 캐시를 조합한 경우와 마찬가지로 드라우지 상태에 놓이는 블록의 수가 많아져 정적 에너지가 추가로 감소한다. 하지만 이 경우에도 순차적 캐시에서 1싸이클, 드라우지 캐시에서 1싸이클의 손해를 보게 되어, 저전력 모드에 있는 블록에 접근했을 경우 총 2싸이클의 손해를 보기 때문에 성능의 감소 폭이 커 실제 구현을 생각하기에는 어려움이 있다.

필터 캐시, 순차적 캐시, 드라우지 캐시 기법을 모두 조합한 경우, 각각의 모듈들은 독립적으로 작동하기 때문에 서로가 가진 에너지 감소 메커니즘은 모두 작동하고 그로 인해 큰 에너지 감소를 얻을 수 있다. 하지만 이 경우 각각의 기법에서 1싸이클씩, 총 3싸이클의 손해를 보게 된다.

본 논문에서는 필터 캐시, 순차적 캐시 그리고 드라우지 캐시 기법을 모두 조합하여 그들이 가진 에너지 감소 메커니즘을 최대한 활용하면서 셋 중 하나의 기법을 적용시켰을 때 가지는 1싸이클의 손해만 입는 새로운 조합법을 연구하였다.

이 조합법을 앞 장에서 제안한 선택적 워드 접근 기법과도 조합하여 동적 전력 소모의 추가적 감소와 워드 단위의 드라우지 캐시 기법으로 정적 전력 소모의 추가적 감소를 도모하였다.

4.2 관련 연구와 제안된 연구의 구현

이 장에서는 본 논문에서 구현하려는 필터 캐시, 순차적 캐시, 드라우지 캐시에 대한 새로운 조합법을 구현하기 위해 그들 간의 조합으로 발생하게 되는 캐시 구조와 필수 경로(Critical Path)의 변화로 생기는 캐시 접근 시간, 그리고 동적·정적 전력의 이득 변화량과 발생하는 오버헤드에 대해 각각 살펴보도록 하겠다.

4.2.1 전통적 L1 기준 캐시

그림 4.1 은 전통적 기준 캐시에 메모리 요청이 접근하는 모습을 나타낸다. 명령어가 전달되면, 가상 태그는 TLB(Translation Lookaside Buffer)에 접근하여 물리 주소 태그로 변환한다. 이러한 태그 변환이 일어나는 동안 인덱스에 해당하는 주소 부분이 L1 태그 어레이와 데이터 어레이에 H-V 네트워크를 통해 전달되고, 변환이 끝난 태그가 L1 태그 어레이에서 적중을 판단한다. 이 과정에서 가장 긴 접근 시간을 가지는 필수 경로는 L1 데이터 어레이에의 접근이다.

32kB, 4웨이, 32B 블락 크기, 32nm 공정기술을 가정하였을 때, CACTI를 통해 캐시가 가지게 되는 인자들의 값은 표 4.1 과 같다. 3Ghz의 프로세서를 가정한 경우, 데이터 어레이 접근 속도는 2 싸이클에 해당하고 이것은 캐시 접근 시간과 동일하다. 앞으로 논의되는 기법들에 대해 기준 값을 이 표를 통해 산정하도록 하겠다.

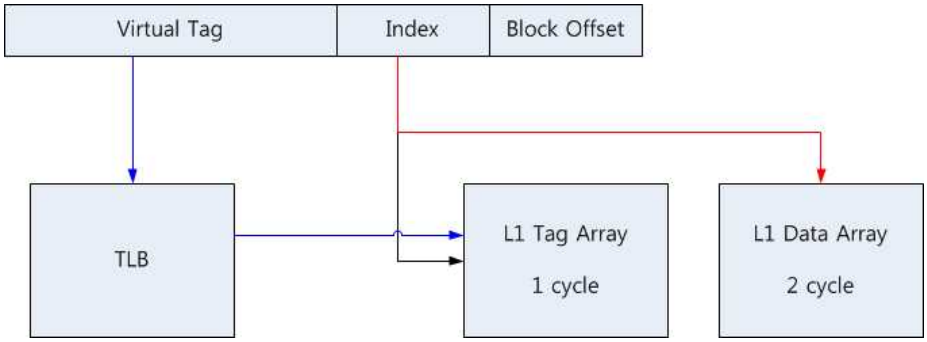

그림 4.1 전통적 기준 캐시의 모습

표 4.1 전통적 L1 기준 캐시의 인자들

전제 접근 시간 (ns)		0.453075
접근 당 동적 에너지 (nJ)		0.047674595
정적 전력 (mW)		20.1058
접근 시간 분석		
태그 어레이 (ns)		0.285265
데이터 어레이 (ns)		0.453075
에너지 분석		
태그 어레이	동적 에너지 (nJ)	0.001697645
	정적 전력 (mW)	2.93087
데이터 어레이	동적 에너지 (nJ)	0.04597695
	정적 전력 (mW)	18.6582
면적 분석		
태그 어레이 (mm2)		0.0163746
데이터 어레이		0.158728

4.2.2 필터 캐시

그림 4.2 는 기준 캐시에 필터 캐시를 적용한 모습이다. 필터 캐시는 L0 위치에 사용되어 L1 캐시에 접근하기 전, 필수적으로 거치도록 구현되어 있다. 이로 인해 필수 경로가 길어지게 되고, 필터 캐시에서 적중 실패가 되었을 경우 1 싸이클의 손해를 보게 된다. 필터 캐시를 64개의 엔트리, 직접 사상, 32B 블록 크기(2kB)로 가정했을 경우, 기준 캐시와 싸이클과 에너지는 다음과 같이 비교된다.

필터 캐시에 적중할 경우, 1 싸이클 동안 메모리 요청을 해결할 수 있기 때문에 기준 캐시보다 1 싸이클 빠르지만 필터 캐시에 적중 실패 했을 경우 기준 캐시보다 1 싸이클 느리게 되어 필터 캐시에의 적중률에 의해 그 성능이 결정된다. 수식으로 표현해보면 다음과 같다.

$$T_{fil\_avg} = R_{fil\_hit} * (1\ cycle) + R_{fil\_miss\_L1\_hit} * (3\ cycle) + R_{L1\_miss} * T_{mem}$$

(T_{fil_avg} : 필터 캐시의 평균 접근 시간,
R_{fil_hit} : 필터 캐시 적중률,
$R_{fil\_miss\_L1\_hit}$: 필터 캐시 적중 실패 후의 L1 캐시 적중률,
R_{L1_miss} : L1 적중 실패율,
T_{mem} : 하위 단계 저장장치 접근 시간)

T_{mem} 위치에는 하위 단계 저장장치 접근 시간이 오는 것이 타당하나, 본 연구에서는 L2 캐시를 배재하였기 때문에 메모리 접근 시간으로 계산하였다. 모의실험 환경에서 이 값은 200 싸이클이다. 응용 프로그램과 필터의 크기에 따라 R_{fil_hit} 값은 달라지지만 60~85% 정도의 적중률을 보이고 있고, 기준 캐시와 L1이 달라지지 않았으므로 R_{L1_miss} 값은 일정하다. 기준 캐시의 경우 SimpleScalar와 SPEC 2000을 사용해 R_{L1_miss} 값을 구해보면 0~10% 정도의 적중 실패율을 보이고 $R_{fil_miss_L1_hit}$은 다음과 같이 구할 수 있다.

$$R_{fil\_miss\_L1\_hit} = 1 - R_{L1\_miss} - R_{fil\_hit}$$

이를 통해 T_{fil_avg}를 계산해 보면, 기준 캐시의 경우 응용 프로그램에 따라 2.2~21.8, 필터 캐시의 경우 1.5~21.5 정도의 값을 가지게 된다. 이를 토대로 L1 기준 캐시의 접근 시간을 2 싸이클로 산정하면, 필터 캐시의 존재가 성능의 감소가 아닌 증가를 가져올 수 있다는 사실을 알 수 있다.

에너지의 경우, 필터 캐시에 적중되었을 경우에는 L1 캐시의 동적 전력 소모만큼 이득을 본다. 하지만 적중 여부와 관계없이 필터 캐시의 동적과 정적 전력 모두의 소모만큼 손해를 본다. 전술한 필터 캐시의 경우, 접근 당 동적 에너지 소모가 0.008nJ로 기준 캐시의 18.4% 정도이고 정적 전력 소모가 1.49mW로 기준 캐시의 7.4% 정도이다. 1회 접근 당 발생하는 동적 에너지의 크기는 다음과 같이 구할 수 있다.

$$E_{dyn\_cache} = R_{fil\_hit} * E_{dyn\_fil} + (1 - R_{fil\_hit}) * (E_{dyn\_fil} + E_{dyn\_L1})$$
$$= E_{dyn\_fil} + (1 - R_{fil\_hit}) * E_{dyn\_L1}$$

(E_{dyn_fil} : 필터 캐시의 접근 당 동적 에너지 소모량
E_{dyn_L1} : L1 캐시의 접근 당 동적 에너지 소모량)

T_{fil_avg}를 계산한 시뮬레이터 결과 값을 통해 추산해본 결과 필터 캐시를 적용했을 시에, 평균적으로 읽기 접근 한 개 당 67.8%의 전력 소모를 보이는 것을 알 수 있다.

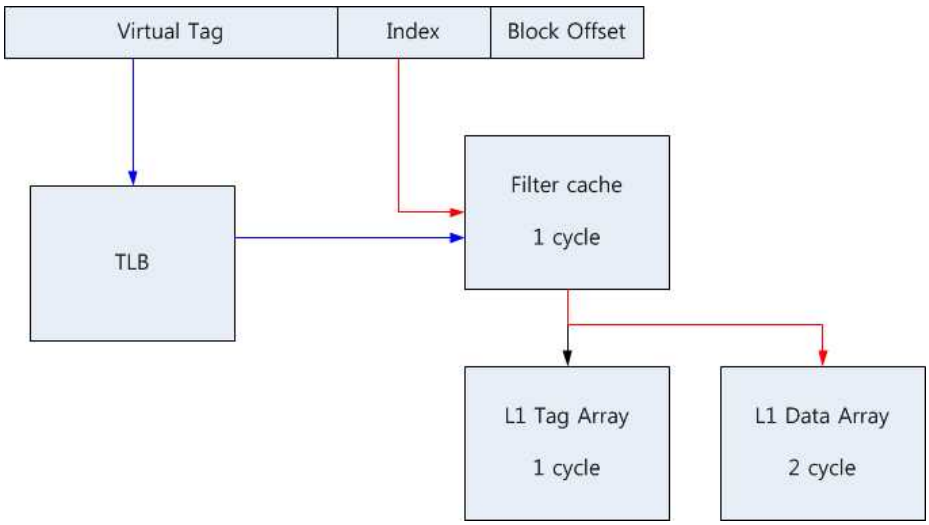

그림 4.2 필터 캐시의 모습

4.2.3 동시 접근 기술을 사용한 필터 캐시

마이크로프로세서 클럭 수의 증가로 인해 기존 논문들에서 1 싸이클로 가정하였던 L1 캐시 접근 시간은 2 싸이클 이상으로 늘어나게 되었다. 그에 따라 필터 캐시가 처음 제안되었던 1997년에는 시스템의 성능을 떨어뜨렸으나, 4.2.2 장에서 언급한 바와 같이 최근의 기술 추세에 의하면 성능을 증가시킬 수 있다는 것을 볼 수 있다.

L0 캐시의 삽입이 바로 이러한 성능 증대의 목표를 가진 연구로, L0 캐시와 필터 캐시는 구조는 비슷하지만 목표와 결과는 상이하다. 그림 4.3과 같이 필터 캐시와 L1 캐시에 병렬적 동시 접근 기술을 사용한 것이 L0 캐시이며, 필터 캐시에 적중될 경우 성능 상에서 1 싸이클의 이득을 볼 수 있지만 병렬적으로 L1 캐시에 접근하기 때문에 필터 캐시의 에너지 소모만큼 손해를 보게 된다.

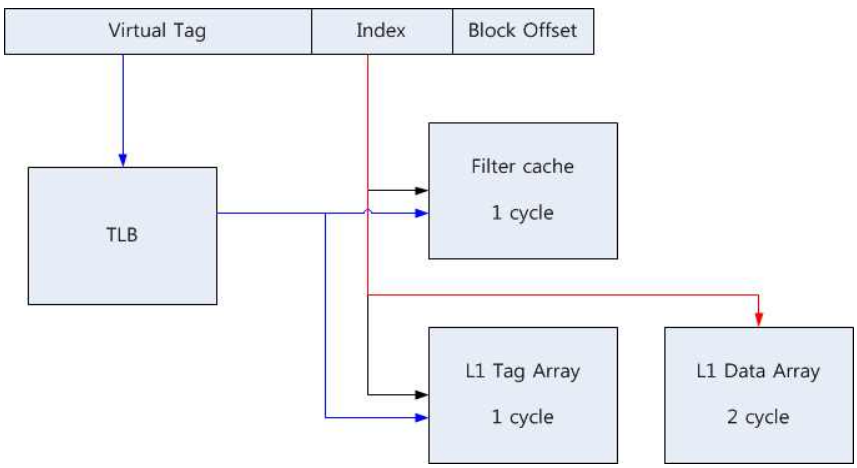

그림 4.3 L1 캐시와 필터 캐시를 동시에 접근하는 모습

4.2.4 순차적 캐시

그림 4.4는 위상 캐시, 혹은 순차적 캐시(Phase Access Cache, Serialized Cache, Sequential Cache)로 불려온 캐시의 모습이다. 이러한 순차적 캐시는 태그 어레이에 먼저 접근한 후, 데이터 어레이에 접속하는 기법을 사용하며, 모든 접근에 대해 1 싸이클의 손해를 본다. 하지만 태그 어레이에 적중했을 시 n-집합 연관 캐시를 가정했을 때 1/n 만큼의 데이터 어레이의 동적 에너지만을 사용하면 되고, 태그 어레이에 적중하지 않았을 시 L1 데이터 어레이에 접근하지 않아 에너지 감소에 효과적이다.

보통 이 기법은 크기가 크고 집합 연관도가 높으며 상대적으로 시간에 대한 제한도가 여유 있는 하위 저장단계 메모리에서 많이 사용된다. L1 캐시의 경우, 집합 연관도가 작고 시간에 대한 제한사항이 크기 때문에 특별한 알고리즘 없이는 이러한 순차적 캐시를 적용시키기 힘들다.

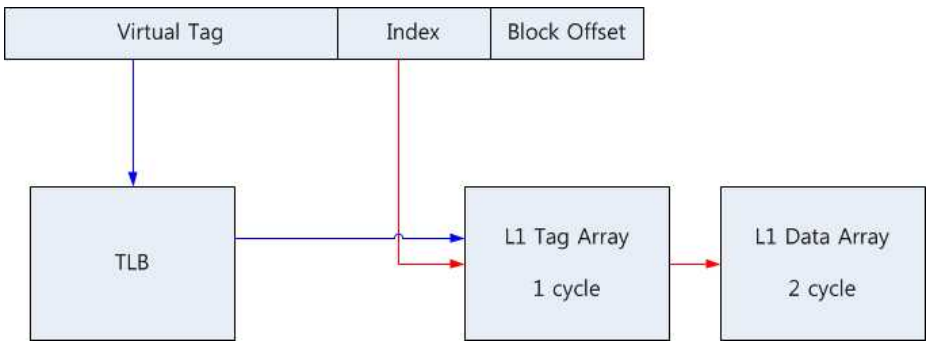

그림 4.4 순차적 캐시의 모습

4.2.5 병렬적 L1 태그 접근 기법과 필터 캐시를 사용한 순차적 캐시

필터 캐시를 통해 순차적 캐시에서 발생하는 시간 지연을 숨기며 L1 캐시에 적용하면, 그림 4.5와 같다. 필터 캐시에서 적중이 발생하면 1 싸이클 만에 메모리 요청이 완료되고, 필터 캐시로의 메모리 요청과 동시에 병렬적으로 L1 태그에 같은 요청이 전달된다. 필터 캐시에의 접근이 적중될 경우 L1 데이터 어레이로의 접근은 차단되고, 필터 캐시에의 접근이 실패하고 L1 태그 어레이에서의 접근도 실패할 경우에도 L1 데이터 어레이로의 접근은 차단된다. 필터 캐시에의 접근 실패와 L1 태그 어레이에서의 접근 성공 시에 L1 데이터 어레이로의 접근이 발생하고 이 때 소모되는 동적 에너지는 기준 캐시 데이터 어레이 동적에너지의 1/n, 이 경우에는 4-웨이 집합연관 캐시이기 때문에 1/4이 된다.

상대적으로 크기가 작아 동적 에너지가 작은 L1 태그 어레이로 병렬적 접근을 하여 순차적 캐시와 필터 캐시를 단순하게 조합하였을 때 발생하는 2 싸이클의 시간 지연을 1 싸이클로 줄였다. 따라서 이 경우의 평균 읽기 요청 시간은 앞에서 살펴본 필터 캐시에서의 요청 시간과 동일하고, 읽기 접근이 발생하고 있을 때의 동적 에너지는 다음 식과 같이 나타난다. 해당 식을 통해 대략적인 계산을 해보면 읽기 접근 한 개당 평균 49.5%의 에너지 소모를 보이는 것을 알 수 있다.

$$E_{tot\_cache} = R_{fil\_hit}*(E_{dyn\_fil} + E_{dyn\_L1\_Tag}) +$$

$$(1 - R_{fil\_hit})*(E_{dyn\_fil} + E_{dyn\_L1\_Tag} + \frac{1}{4}E_{dyn\_L1\_Data})$$

$$= Ed_{yn\_fil} + E_{dyn\_L1\_Tag} + (1 - R_{fil\_hit})*E_{dyn\_L1\_Data}/4$$

($E_{dyn\_L1\_Tag}$: L1 태그 어레이의 동적 에너지

$E_{dyn\_L1\_Data}$: L1 데이터 어레이의 동적 에너지)

필터 캐시만 적용했을 때의 67.8%보다 18%의 추가 에너지 소모 감소만을 보이는 이유는, L1 태그 어레이가 L1 데이터 어레이에 비해 3.7% 수준의 동적 에너지만을 가지고 있지만, 필터 캐시에 비하면 그 크기가 19.4%에 해당하는 동적 에너지를 가지기 때문이다.

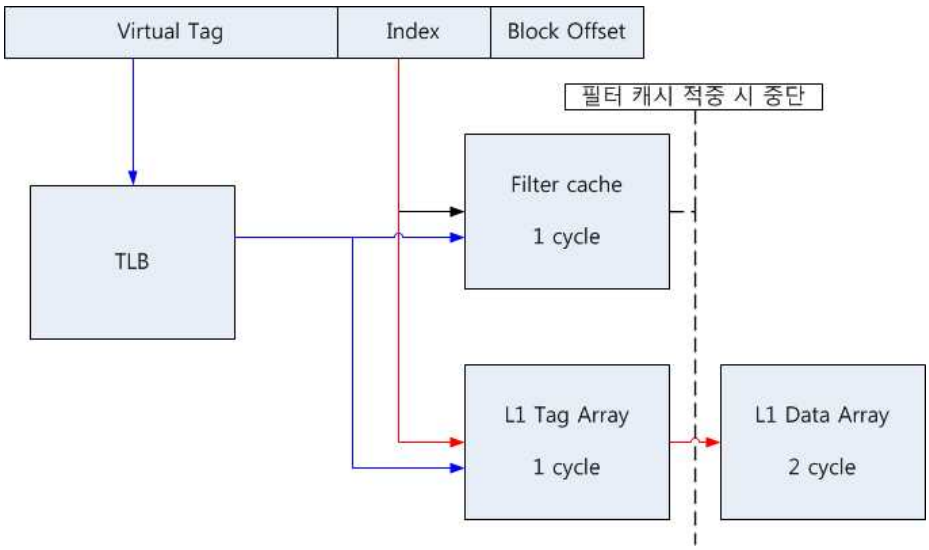

그림 4.5 필터 캐시가 사용된 순차적 캐시에 병렬적 L1 태그 접근 기법을 사용한 모습

4.2.6 드라우지 캐시

드라우지 기법을 적용시키기 위한 정책은 전술한 바와 같이 단순 정책, 미접속 정책, 가장 최근에 접근한 블록은 드라우지 모드에 빠지지 않는 MRO 정책 등 여러 가지가 있으나, 본 연구에서는 2000 싸이클마다 모든 데이터 어레이가 저전력 모드에 돌입하는 단순 정책을 사용한다. 이 외에 태그 어레이를 드라우지 모드로 동작시키는 방법도 있으나 본 연구에서는 배재한다.

드라우지 기법을 적용하려면, 집합 연관 캐시에서 셋당 한 비트 씩을 추가적으로 유지하는 오버헤드가 필요하다. 그림 4.7 과 같은 형태의 드라우지 캐시에서 SPEC 2000의 구동을 시켜본 결과, 5~10% 정도의 성능 감소와 85% 정도의 정적 전력 감소를 얻어낼 수 있었다. 성능 감소로 인한 시간 지연으로 실이득 전력 감소는 80% 정도로 관측되었다.

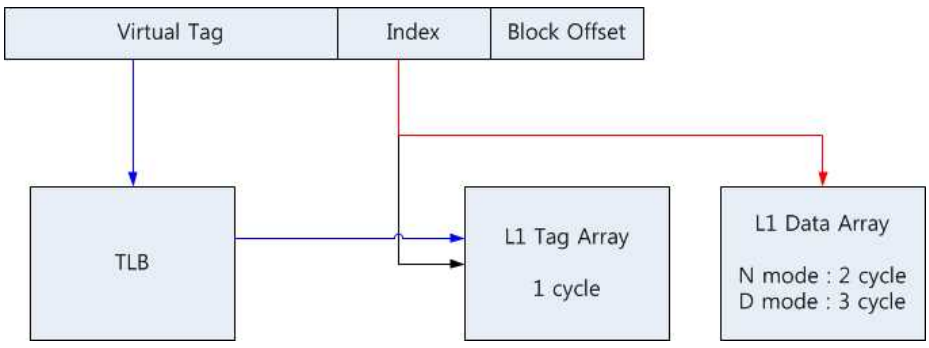

그림 4.6 드라우지 캐시의 모습

4.2.7 필터 캐시를 사용한 드라우지 캐시

필터 캐시와 드라우지 캐시를 혼용한 형태의 명령어 캐시는 2008년도에 Giorgi에 의해 제안된 바 있다. 그림 4.8과 같은 형태의 캐시 모습을 하고 있고, 주 목적은 필터 캐시를 사용하여 L1 데이터 어레이의 접근을 차단, 드라우지 모드에 있는 시간을 늘려 정적 에너지 감소를 도모하는 기법이다. [GB+08]

이 기법을 사용하여 필터 캐시에 적중되었을 경우에는 메모리 요청이 1 싸이클, 필터 캐시에 적중 실패가 발생하고 정상 전압 모드인 블록에 적중되었을 경우에는 3 싸이클, 저전력 모드인 블록에 적중되었을 때에는 4 싸이클의 시간이 걸리는 것을 확인할 수 있다. 따라서 필터 캐시의 적중률에 의해 그 성능이 크게 달라진다.

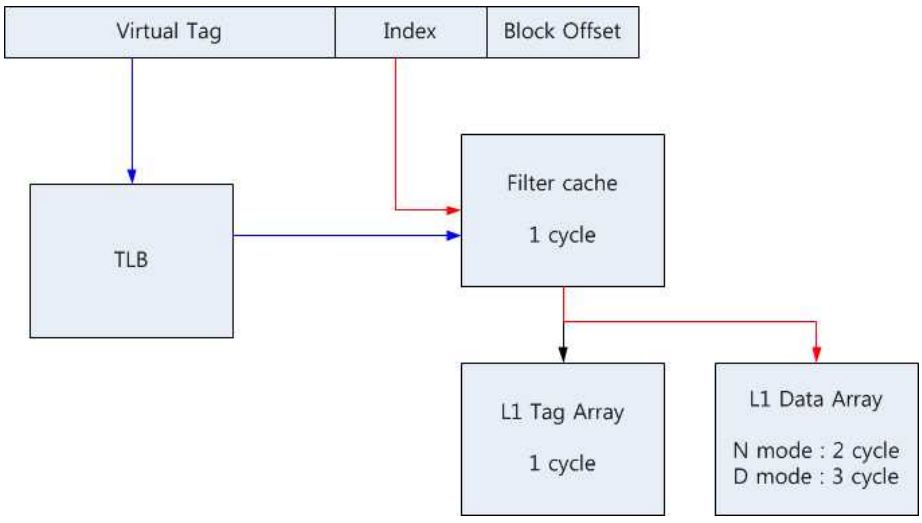

그림 4.7 필터 캐시가 사용된 드라우지 캐시의 모습

4.2.8 병렬적 동시 접근을 이용한 드라우지·필터 캐시

그림 4.8 은 앞 절에서 살펴본 드라우지·필터 캐시의 성능 저하를 막고자 필터 캐시에 접근함과 동시에 L1 드라우지 캐시에 접근하는 기법이다. 이 기법을 사용할 경우, 필터 캐시에 적중했을 시 1싸이클, 적중하지 않았을 시에 2~3 싸이클이 걸려 드라우지 캐시의 장점을 살리며 필터 캐시에서의 시간적 이득을 가져갈 수 있으나, 모든 접속에 대해 L1 캐시의 저전력 모드를 깨우기 때문에 필터 캐시의 필터링에 대한 의미가 퇴색되고, 필터 캐시에 의해 에너지 감소는 전혀 보지 못하고 오버헤드로서만 동작하는 것을 볼 수 있어 의미 없는 조합법임을 알 수 있다.

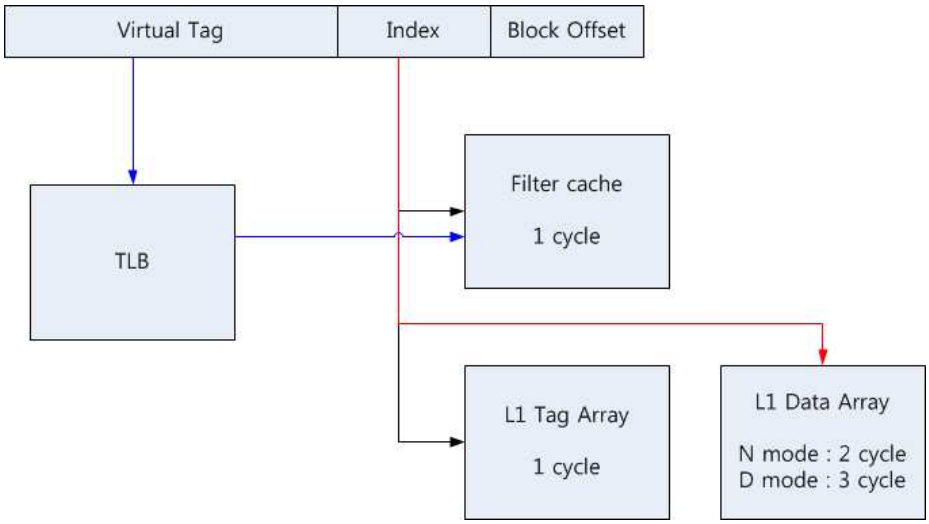

그림 4.8 드라우지 캐시와 필터 캐시가 모두 사용되었을 때 L1 캐시와 필터 캐시를 동시에 접근하는 모습

4.2.9 필터 캐시를 사용한 순차적 드라우지 캐시

그림 4.9 은 필터 캐시와 드라우지 캐시, 순차적 캐시가 모두 사용된 모습이며 그림 4.10 은 해당 구현에 대한 메모리 요청 순서도이다. 필터 캐시에 의한 L1 접근의 필터링과, 순차적 캐시에 의한 데이터 어레이의 웨이 필터링 그리고 드라우지 캐시의 정적 전력 감소, 순차적·필터 캐시에 의한 드라우지 모드 시간의 연장 효과로 최대의 전력 감소 효과를 볼 수 있으나, 필터 캐시에서 적중 실패하고 L1 캐시에 적중했을 경우 캐시 접근 시간이 일반 모드일 경우 4 싸이클, 저전력 모드일 경우 5 싸이클로 늘어나는 것을 볼 수 있다. 이러한 조합법을 사용할 경우, 모든 저전력 기법에 의한 에너지 감소효과를 관찰할 수 있지만 최악의 경우 기준 캐시보다 2.5배 느린 접근 시간을 가지기 때문에 사용하기 힘들다.

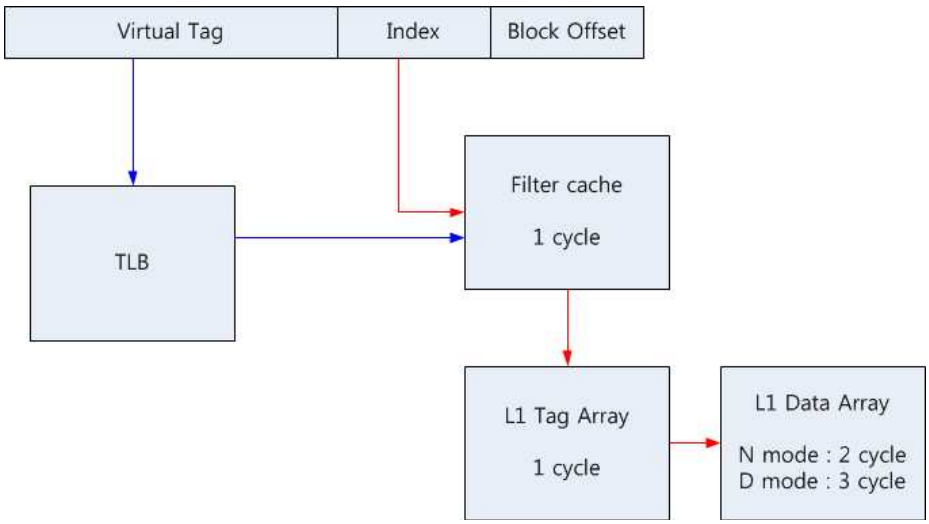

그림 4.9 필터 캐시, 드라우지 캐시, 순차적 캐시가 모두 사용된 모습

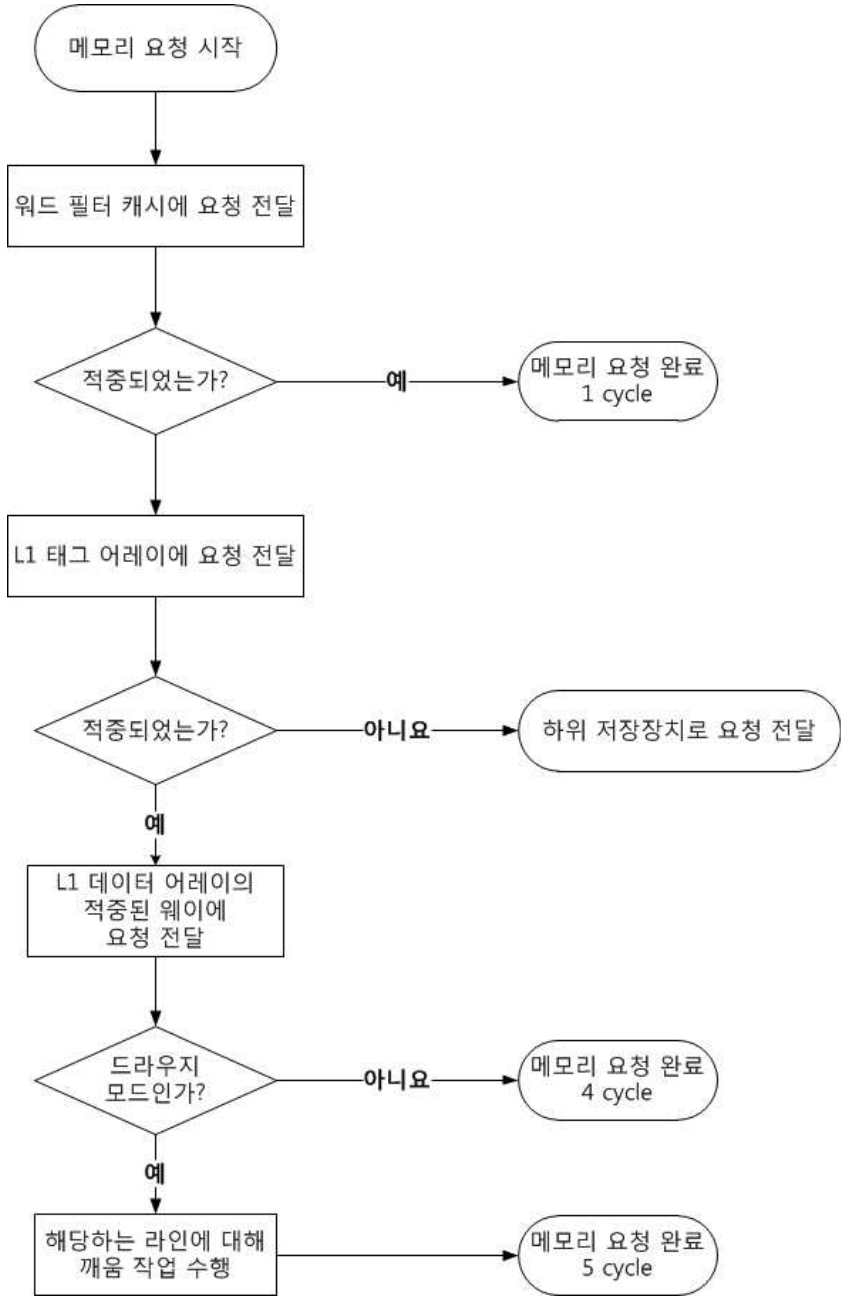

그림 4.10 필터 캐시, 드라우지 캐시, 순차적 캐시가 모두 사용되었을 때 메모리 요청에 관한 순서도

4.2.10 L1 태그에 병렬적 접근과 깨움 동작을 하는 순차적 드라우지·필터 캐시

앞 절에서 살펴본 바와 같이 순차적 캐시와 드라우지 캐시, 그리고 필터 캐시를 모두 조합할 경우 전력 감소 효과를 최대로 할 수 있지만, 성능의 감소폭도 커지는 것을 볼 수 있다. 이를 그림 4.11 와 같이 구성하여 성능의 감소폭을 최소화 하면서 최대한의 전력 이득을 가져갈 수 있도록 하였다.

L1 데이터 어레이에 비해 동작 전력이 작은 L1 태그 어레이를 필터 캐시와 병렬적으로 접근시키고, 그와 동시에 L1 데이터 어레이에 깨움 신호를 보내어 해당 셋이 저전력 모드에 있다면 태그 어레이를 접근하는 동안 깨운다. 그 후 필터 캐시가 적중하거나 L1 태그 어레이에서 적중 실패가 발생할 경우 L1 데이터 어레이에는 접근하지 않는다.

이러한 구조를 가지게 된 캐시는 필터 캐시에서와 동일한 성능의 이득을 얻음과 동시에, 필터 캐시에서와 L1 태그 어레이에서의 필터링에 의해 L1 데이터 어레이에 접속을 제한하여 동적 에너지 소모에서 이득을 가져간다. 또한, 드라우지 캐시를 접목시켜 드라우지 모드로 있을 때 정적 전력 소모에서 이득을 가져갈 수 있다.

단순하게 세 기법을 조합만 했을 때에 비해, L1 태그 어레이에의 접근이 병렬적으로 일어나고 드라우지 모드에서 깨우는 과정이 필

터링되지 않아 에너지의 손해를 보게 되지만, 성능 상에서 단 한 개의 기법만을 조합했을 때와 같은 오버헤드를 가지게 되는 것은 주목할 만한 일이다. 추가적으로 전술한 바와 같이 최근의 하드웨어를 가정할 경우, 필터 적중 실패가 발생했을 때 싸이클 수는 증가하지만 응용 프로그램 전체에서 보았을 때에는 오히려 성능이 증가하는 것을 살펴본 바 있다.

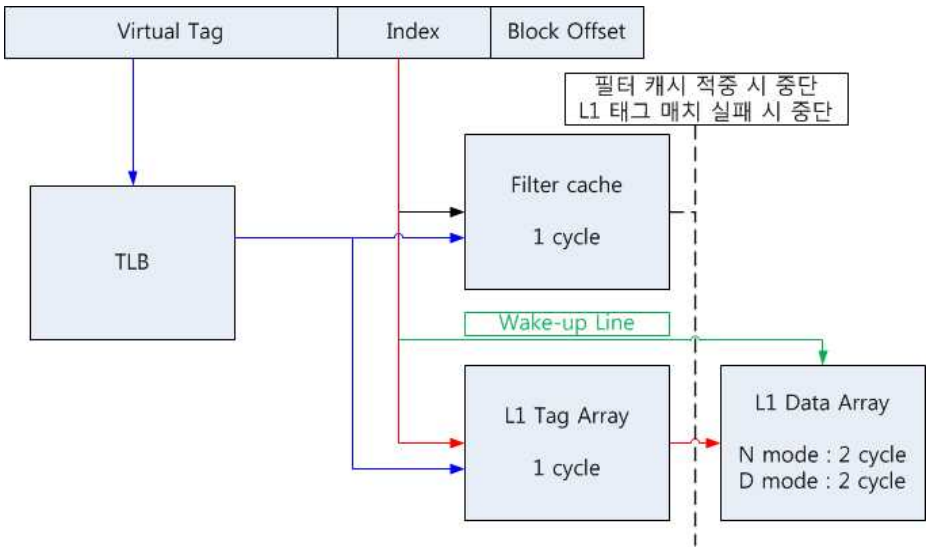

그림 4.11 드라우지·필터·순차적 캐시가 모두 사용되었을 때 병렬적 L1 태그 접근 기법을 사용한 모습

4.3 선택적 워드 접근 기법과의 조합 구현

앞 절에서 제안한 병렬적 L1 태그 접근 기법을 사용한 드라우지·필터·순차적 캐시에 앞 장에서 제안한 선택적 워드 접근 기법을 조합할 수 있다. 선택적 워드 접근 기법을 조합할 경우, 필터 캐시의 저장 단위는 8B 1워드가 되어야 하며 드라우지 기법을 워드 별로 잘게 나누어 (fine-grain) 적용해야 한다. (그림 4.12)

필터 캐시의 저장 단위를 32B에서 8B로 줄이게 되면, 같은 용량을 전제했을 시에 엔트리(entry)의 개수가 4배로 늘어날 수 있고, 같은 엔트리를 전제할 경우, 용량이 1/4로 줄어들 수 있다. 엔트리의 개수가 늘어날 경우, 데이터 캐시처럼 공간적 연관성이 적은 캐시에 대해 더 좋은 적중률을 보일 수 있다.

그림 4.13에서 보는 것처럼 워드 단위로 깨움 작업을 하기 위해서는 전 장에서 설명한 것과 같이 블록 오프셋의 상위 2비트를 추가적으로 L1 데이터 어레이에 전송하여야 한다. 그리고 기존에 셋 별로 1 비트씩 추가되었던 드라우지 비트를 이제는 워드 별로 1비트씩 추가하여 관리하여야 한다. 이렇게 잘게 나누어 전력 모드를 관리하게 되면, 오버헤드에 해당하는 추가 비트가 늘어나지만 블록 별로 관리할 때에 비해 드라우지 모드에 놓이는 비율이 높아진다.

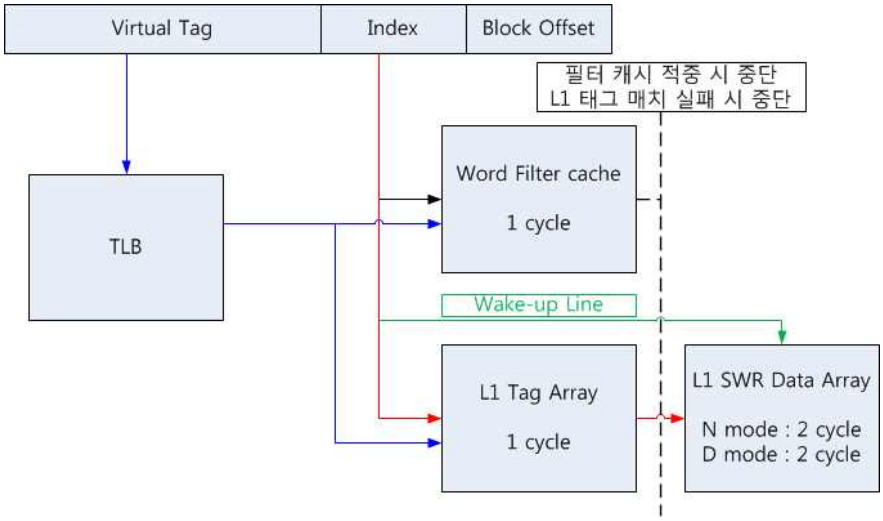

그림 4.12 선택적 워드 접근 기법과 조합한 모습

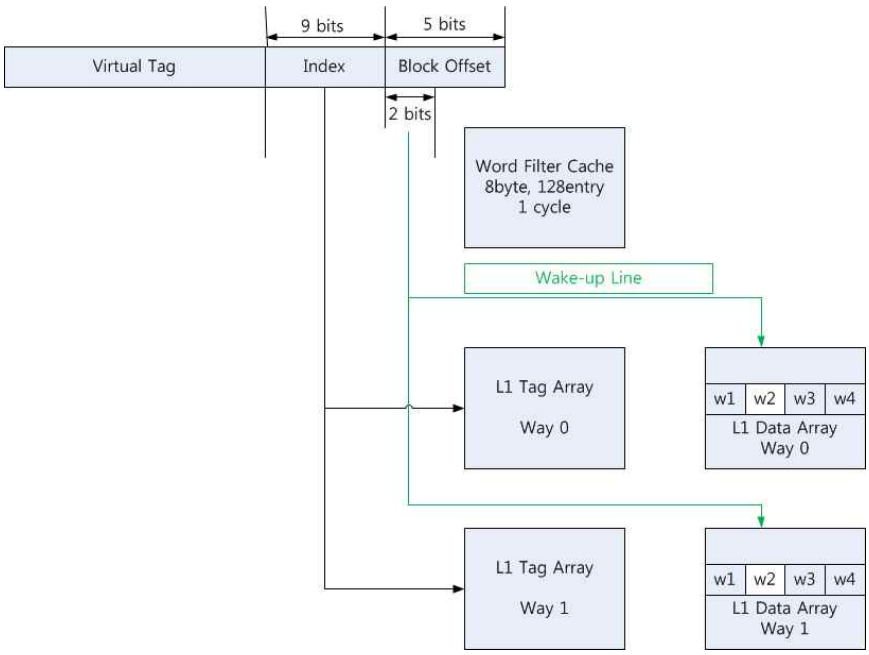

그림 4.13 깨움(Wake-up) 작업을 위한 인덱스 전달 과정

그림 4.14는 워드 필터를 사용한 순차적·선택적 워드 접근 드라우지 캐시의 메모리 요청 순서도이다. 이 순서도를 통해, 필터 캐시에 드라우지 캐시와 순차적 캐시를 추가하였음에도 불구하고 필터 캐시 하나만을 구현했을 때와 접근 시간이 동일한 것을 알 수 있다.

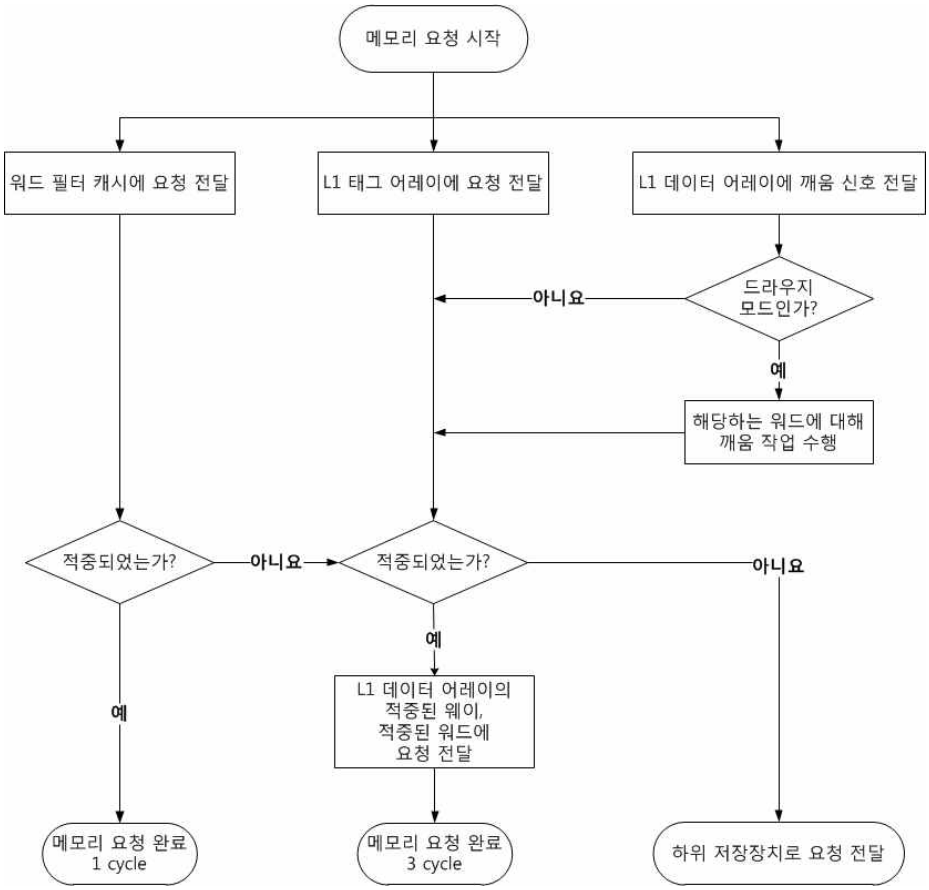

그림 4.14 워드 필터를 사용한 순차적·선택적 워드 접근 드라우지 캐시의 메모리 요청 순서도

이러한 배치를 통하여 성능은 필터 캐시와 동일하고, 읽기 동작을 할 때의 동적 에너지 소모는

$$E_{tot\_cache} = R_{fil\_hit}*(E_{dyn\_fil} + E_{dyn\_L1\_Tag}) + (1 - R_{fil\_hit})*(E_{dyn\_fil} + E_{dyn\_L1\_Tag} + \frac{1}{16}E_{dyn\_L1\_Data})$$

$$= E_{dyn\_fil} + E_{dyn\_L1\_Tag} + (1-R_{fil\_hit})*E_{dyn\_L1\_Data}/16$$

와 같이 나타나게 됨을 알 수 있다. 이를 통해 동적 에너지 소모를 계산해보면, 기준 캐시에 비해 평균 33.28%의 동적 에너지를 소모하는 것을 알 수 있다. 이 수치는 선택적 워드 접근 캐시의 38.75%, 필터 캐시의 67.77%, 병렬적 L1 태그 접근 기법을 사용한 순차적·필터 캐시의 49.50%에 비해 더 좋은 결과인 것을 확인할 수 있다.

드라우지 캐시가 같이 구현되어 있기 때문에 정적 전력 감소 효과도 추가적으로 얻을 수 있는데, 워드 단위로 잘게 나누어진 관리를 하기 때문에 얻는 이득과, 전통적 드라우지 캐시에 비해 늘어난 3비트를 관리하는 손해를 종합해서 생각해야 한다. 또, 필터 캐시의 존재로 인해 총 수행시간이 줄어들어 추가적인 정적 에너지 감소를 가져올 수 있다.

제 5장 성능 평가 및 결과

이번 장에서는 실제 모의실험을 통해 본 논문에서 제안한 선택적 워드 접근 기법(이하 SWR 캐시)과 워드 필터가 사용된 순차적·선택적 워드 접근 드라우지 캐시(이하 SSDF 캐시)의 성능과 에너지 면에서의 향상 정도를 살펴본다.

5.1 실험 환경

본 논문에서 제안된 기법들을 성능 평가하기 위해 CACTI 6.5 전력 모델을 사용하여 필터 캐시, L1 캐시 그리고 L2 캐시를 모의 설계하였다. 이 캐시들의 구동 정보를 알기 위해 구동 기반 시뮬레이터인 SimpleScalar를 수정하여 사용하였으며, 모의실험에서 사용된 벤치마크 프로그램은 SPEC(Standard Performance Evaluation Corporation)에서 제공하는 SPEC 2000을 사용하였다. SPEC 2000은 정수형 프로그램 그룹인 CINT와 실수형 프로그램 그룹인 CFP로 구성된다. CINT 프로그램 그룹은 정수형 프로그램으로써 정형화되지 않은 코드의 응용 프로그램이 주를 이루며 분기 예측이 힘들고 메모리 접근 주소 또한 예측하기 쉽지 않다. 반면에 CFP 프로그램 그룹은 과학 계산 형태의 응용 프로그램이 주를 이루며, 이들은 매우 정형화되어 있는 형태의 코드를 가진다. 이 같은 경우, 분기 예측의 정확도가 매우 높으며 메모리 접근 주소 또한 예측하기 쉽다. 본 논문의 서론의 연구 배경에서 언급되었던 약한 계산 응용 프로

그램(Non-compute Heavy Program)이 CINT, 강한 계산 응용 프로그램(Compute Heavy Program)이 CFP에 해당한다. 본 논문에서는 CINT 프로그램 그룹과 CFP 프로그램 그룹 모두를 모의실험에 사용하였다. 모의실험에 사용된 벤치마크 프로그램들은 각각 gcc 컴파일러를 이용하여 ALPHA Binary 형태로 컴파일 되었다.

표 5.1과 5.2는 각각 본 논문에서 사용된 CINT 프로그램 그룹과 CFP 프로그램 그룹의 벤치마크 프로그램들을 간략한 설명으로 소개하고 있다.

표 5.1 CINT 프로그램 그룹

BenchMark	Description
gzip	Data compression utility
vpr	FPGA circuit placement and routing
mcf	Minimum cost network flow solver
crafty	Chess program
parser	Natural language processing
eon	Ray tracing
gap	Computational group theory
vortex	Object Oriented Database
bzip2	Data compression utility
twolf	Place and Route Simulator

표 5.2 CFP 프로그램 그룹

Benchmark	Description
swim	Shallow Water Modeling
mgrid	Multi-grid Solver: 3D Potential Field
applu	Parabolic / Elliptic Partial Differential Equations
mesa	3-D Graphics Library
galgel	Computational Fluid Dynamics
art	Image Recongnition / Neural Networks
equake	Seismic Wave Propagation Simulation
ammp	Computational Chemistry
lucas	Number Theory / Primality Testing
fma3d	Finite-element Crash Simulation
apsi	Meteorology: Pollutant Distribution

선택적 워드 접근 기법의 모의실험을 할 때와 SSDF 캐시의 모의실험을 할 때 실험 환경에 약간 차이가 있다. 표 5.3을 통해 선택적 워드 접근 기법의 모의실험을 할 때의 모의실험 인자에 대해 정의하고, SSDF 캐시의 모의실험을 설명하면서 추가되거나 다른 값을 가지게 되는 인자들에 대해 새로이 정의하도록 하겠다.

표 5.3 모의실험 인자

Parameter		Value
CPU Clock		3 Ghz
Instruction Fetch Queue		2 or 4 or 8
Fetch, Decode Width		2 or 4 or 8
Issue Width		2 or 4 or 8
ROB entries		16
LSQ entries		8
Functional Units(integer)		1 ALU, 1Mult/Div
Functional Units(floating point)		1 ALU, 1Mult/Div
Instruction TLB		128(4 set x 32 entries) entries
Data TLB		128(4 set x 32 entries) entries
Predictor Style		bimod, 2048 entries
BTB entries		2048(4 set x 512 entries) entries
RAS entries		8 entries
Branch Misspredict ion Penalty		3 cycle
L1 Cache		Baseline: Nr. of Mats : 2, Nr. of sub-banks: 2 SWR: Nr. of Mats : 4, Nr. of sub-banks: 1
	I-Cache	32kB, 4 way, 32B line, 2 cycle
	D-Cache	32kB, 4 way, 32B line, 2 cycle
L2 Unified Cache		1MB, 16 way, 64B line, 10 cycle Nr. of Mats: 8, Nr. of sub-banks: 8
Memory Latency		first chunk = 200 cycles inter chunk = 4 cycles
Simulator		10^7 instructions warm up, measure for 10^8 instructions

5.2 선택적 워드 접근 기법 실험 결과

5.2.1 동적 에너지 감소량

전술한 바와 같이, 선택적 워드 접근 기법 캐시에서는 읽기 명령어가 수행될 시 한 개의 매트만을 접근한다. 읽기 접근 한번 당 동적 에너지는 기준 캐시에 비해 L1의 경우 38.7%로 감소되어 61.3%의 이득을 볼 수 있고, L2의 경우 51.5%로 감소되어 48.5%의 이득을 볼 수 있다.

캐시에서 소모되는 동적 에너지 소모량은 다음과 같이 표현된다.

$$E_{dyn} = E_{dyn\_SWR} * C_{Load\_hit} + E_{dyn\_L1} * (C_{tot} - C_{Load\_hit})$$

(E_{dyn_SWR}: SWR 캐시에서 소모되는 접근 당 동적 에너지 소모량,
E_{dyn_L1}: 전통적 L1 캐시에서 소모되는 접근 당 동적 에너지 소모량,
C_{tot}: 총 캐시 접근 횟수
C_{Load_hit}: 읽기 접근 성공 횟수)

그림 5.1 은 각 벤치마크에 대해 기준 캐시 대비 선택적 워드 접근 기법 캐시의 동적 에너지 소모량을 나타낸 것이다. L1 캐시에서는 “art”에서 62.03%의 최소 감소량을 보였고, “mesa”에서 68.51%의 최대 감소량을 보여 평균 67.28%의 동적 에너지 감소량을 보였다. L2 캐시에서는 “swim”에서 34.86%의 최소 감소량을 보였고,

“lucas”에서 49.21%의 최대 감소량을 보여 평균적으로 42.75%의 동적 에너지 감소량을 보였다. 전체 캐시에서는 “art” 벤치마크에서 43.65%로 최소 감소량을 보였고, “twolf”에서 68.15%의 최대 감소량을 보여 평균적으로 61.58%의 동적 에너지 감소량을 볼 수 있었다.

이 결과에서 볼 수 있듯이 L1 캐시에서 L2 캐시에서보다 에너지 소모 감소량의 응용 프로그램 별 변동 폭이 크지 않다. L1 캐시는 L2 캐시에 비해 상대적으로 큰 적중률과 작은 적중률의 표준 편차를 가지고 있기 때문에, L1 캐시에서의 에너지 소모량의 표준 편차가 크지 않다. 선택적 워드 접근 기법은 읽기 접근의 적중 시에만 동적 에너지 감소를 가져올 수 있기 때문에, “art”와 “sixtrack”과 같은 워킹 셋이 크고 메모리 요청 명령어의 비율이 큰 응용 프로그램에서 좋은 결과를 얻기 힘들다.

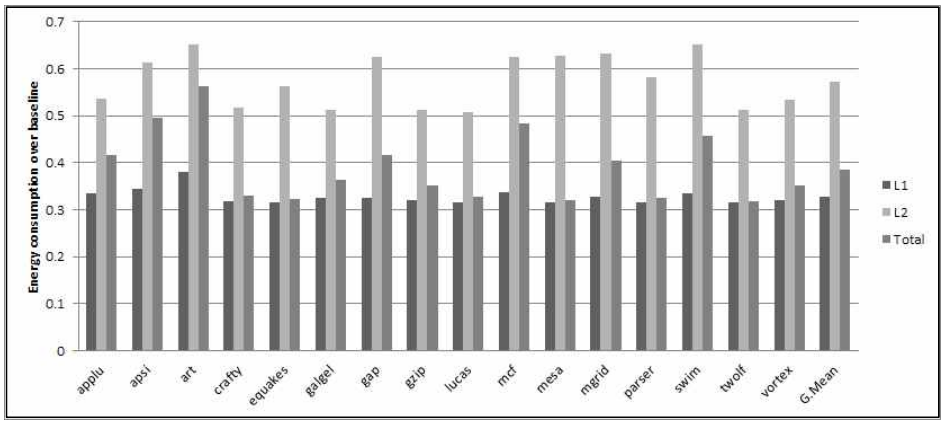

그림 5.1 선택적 워드 접근 기법 캐시의 동적 에너지 소모량

5.2.2 정적 에너지를 고려한 상태에서의 에너지 감소

정적 에너지는 캐시가 동작하는 시간 동안에 소모되는 에너지이다. 캐시의 사이즈가 크고, 동작 시간이 길수록 정적 에너지의 양은 크게 나타난다. 대부분의 동적 에너지 감소 전략들은 성능의 감소를 담보하고 있고, 그로 인해 동작 시간이 늘어나 정적 에너지의 양이 추가적으로 늘어나게 된다. 드라우지 캐시 같은 정적 에너지 감소 전략들도 성능이 감소되기 때문에, 알고리즘 자체의 정적 에너지 감소량에 동작 시간 증가로 인한 정적 에너지 증가량을 같이 계산해야 정확한 정적 에너지 량을 측정할 수 있다. 본 논문에서는 다음 식과 같이 정적 에너지 량을 측정하였다.

$$E_{leak} = P_{leak} * total\ cycle * cycle\ time$$

P_{leak}는 CACTI를 통하여, total cycle은 SimpleScalar를 통하여 구할 수 있으며 cycle time은 모의실험 인자로 3Ghz의 프로세서를 가정하였기에 0.3333(ns) 가 된다.

그림 5.2와 5.3은 L1과 L2에서의 총 소모된 에너지 량의 기준 캐시 대비 비율을 나타낸다. L1 캐시에서는 최소값 27.25%의 감소량을 가진 “art”와 최대값 61.54%의 감소량을 가진 “fma3d”를 관측할 수 있었고, L2 캐시에서는 최소값 0%의 감소량을 가진 “mesa”와 5.24%의 감소량을 가진 “apsi”를 관측할 수 있었다. L1과 L2 캐시는 각각 평균 56.75%, 1.06%의 전체 에너지 감소량을 가진다.

L1에서와 달리 L2에서는 1% 대의 저조한 에너지 감소율을 보이고 있다. L2 캐시의 크기가 크기 때문에 전체 에너지 소모량의 90% 이상을 정적 에너지가 차지하고 있고, 상대적으로 빠른 종료 시각을 가지는 “apsi”, “gcc”, “mcf”와 같은 응용 프로그램에서만 5% 정도의 에너지 감소량을 보이고 있다.

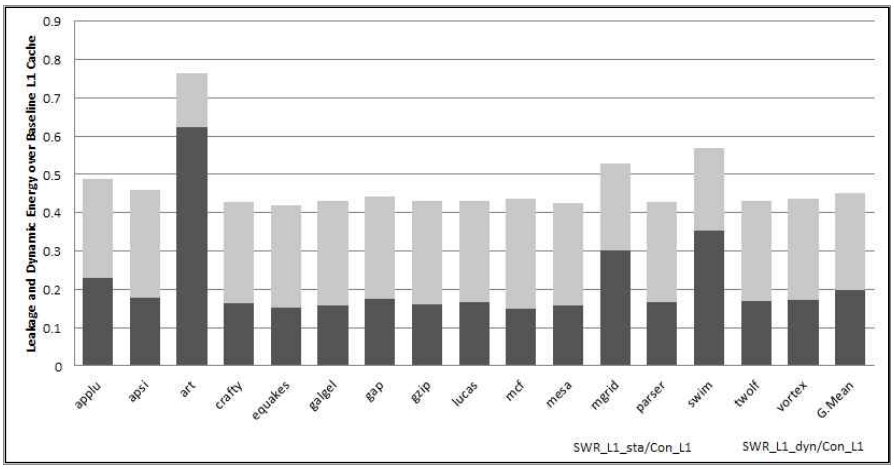

그림 5.2 정적 전력을 고려한 선택적 워드 접근 기법 L1캐시의 에너지

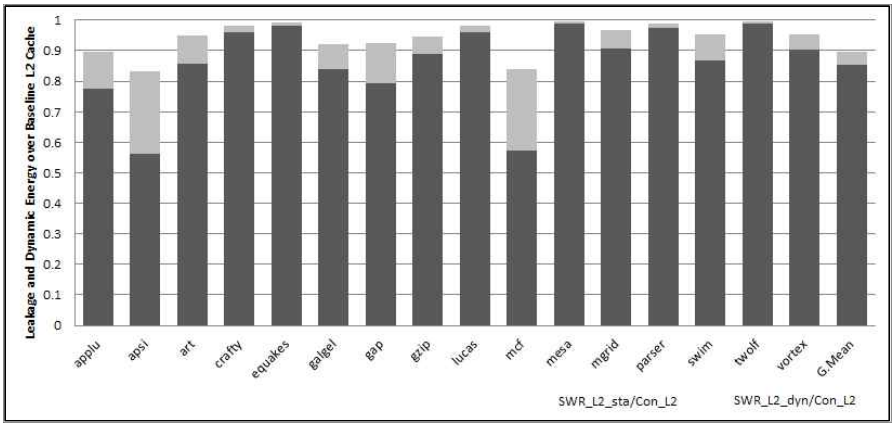

그림 5.3 정적 전력을 고려한 선택적 워드 접근 기법 L2 캐시의 에너지

그림 5.4에서는 L1과 L2 캐시를 합친 전체 캐시의 에너지 소모량을 나타내고 있다. 전체 캐시에서는 “art”의 최소 감소량 2.83%와, “fma3d”의 최대 감소량 16.32%와 함께, 평균 11.71%의 에너지 감소량을 보였다. 앞서 살펴본 그래프들을 종합해 보면, 11.71%의 에너지 감소는 대부분이 L1 캐시의 동적 전력 감소에서 왔다는 것을 알 수 있다. “art”와 “ammp”와 같은 특별히 오랜 동작 시간을 가지는 응용 프로그램을 제외하면, 전체 11.74%의 평균 에너지 감소량을 보인다.

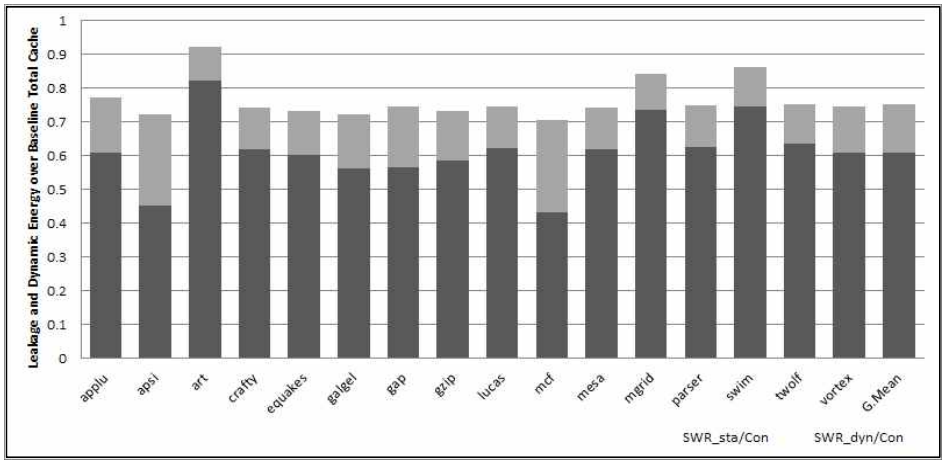

그림 5.4 정적 전력을 고려한 선택적 워드 접근 기법의 전체 캐시의 에너지

5.2.3 워드 버퍼를 가정한 경우 에너지 소모량

워드 버퍼는 L1 캐시의 동적 전력을 추가적으로 감소시킬 수 있는 효율적인 방법으로 그에 관한 논의는 본 논문의 3.2절에서 했다. 그림 5.5는 L1 캐시의 동적 에너지 소모량을 관측한 결과이다. 워드 버퍼를 선택적 워드 접근 캐시에 추가 구현하면 L1 캐시는 21.13%의 에너지만을 소모하여 선택적 워드 접근 기법만 구현했을 때보다 10% 정도의 동적 에너지를 더 감소시킬 수 있다. 이것은 정적 에너지까지 고려한 전체 캐시를 생각했을 때 0.7% 정도를 더 감소시킬 수 있는 량이다.

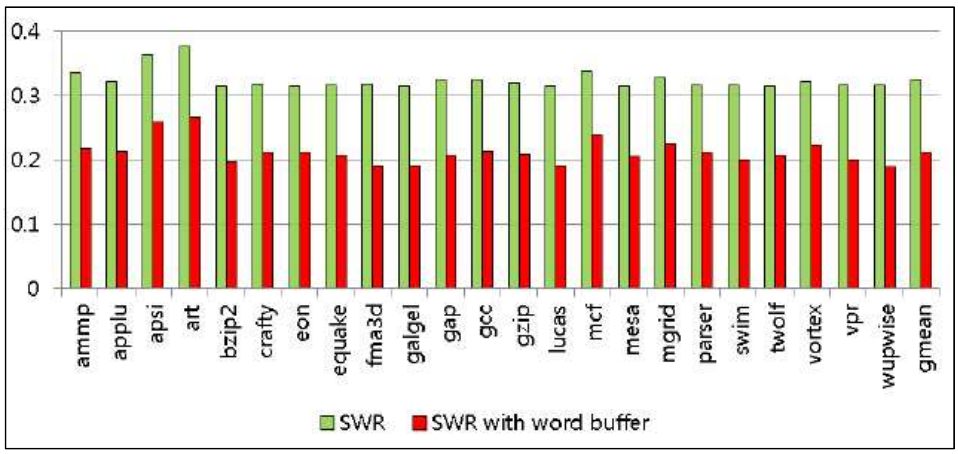

그림 5.5 워드 버퍼가 구현된 SWR L1 캐시의 동적 에너지 소모량

5.2.4 에너지-지연시간 곱

3.4절에서 논의한 바와 같이 명령어 캐시에 선택적 워드 접근 기법을 사용할 경우, 한 번의 접근으로 처리할 수 있는 명령어의 수가 달라지고 그로 인해 명령어 인출 너비가 2인 경우 75%, 4인 경우 98%의 IPC를 관찰하였다.

따라서, 에너지가 감소하지만 지연시간이 증가하는 결과가 나오게 되므로 그를 분석하기 위해 에너지-지연시간 곱을 계산한 결과가 그림 5.6과 같다. 명령어 인출 너비가 2인 경우 에너지-지연시간 곱은 기준 캐시에 비해 평균적으로 33.70% 감소하는 것을 볼 수 있고, 명령어 인출 너비가 4인 경우 평균적으로 22.49% 감소하는 것을 볼 수 있다. 명령어 인출 너비가 2인 경우 IPC가 크게 감소하지만 그에 비례해 에너지도 크게 감소하기 때문에, IPC의 감소폭이 작은 명령어 인출 너비가 4인 경우보다 에너지-지연시간 곱의 감소량이 더 작다.

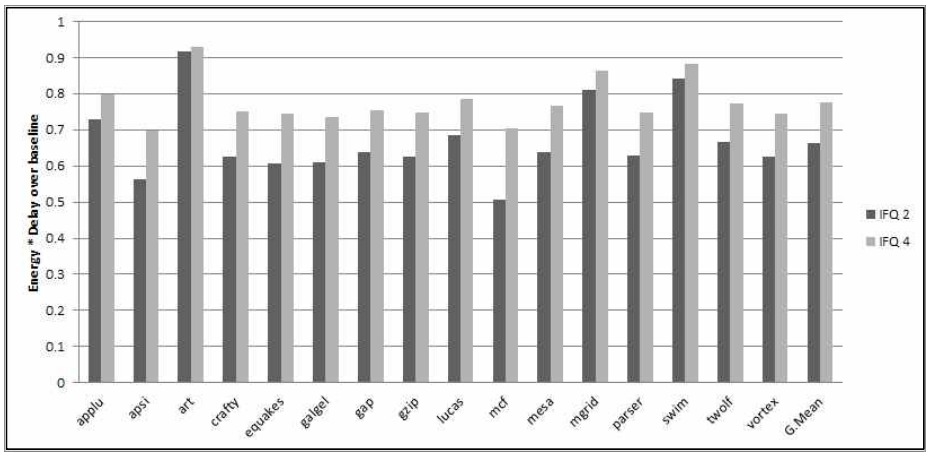

그림 5.6 명령어 인출 너비 변화에 따른 에너지-지연시간 곱

5.3 SSDF 캐시의 실험 결과

워드 필터가 사용된 순차적·선택적 워드 접근 드라우지 캐시(이하 SSDF 캐시)의 구현에 있어서 선택적 워드 접근 기법 캐시(이하 SWR 캐시)와는 다른 캐시 구조를 가지게 된다. 결과를 단순하고 집중적으로 분석하기 위해 L2 캐시를 배재하고, 필터 캐시를 추가한다. 필터 캐시의 구성은 4.2.2 장에서 논한 바와 같이 한다.

5.3.1 필터 캐시의 영향 분석

그림 5.7은 필터 캐시 적중 실패율을 분석한 그래프이다. 공간적 연관성이 큰 명령어 캐시의 경우 적중 실패율이 평균 8.8%가 나왔고, 데이터 캐시의 경우 35.4%의 적중 실패가 발생하였다. 두 캐시를 합하여 생각해 보았을 때에는 14.6%의 적중 실패가 발생하였다.

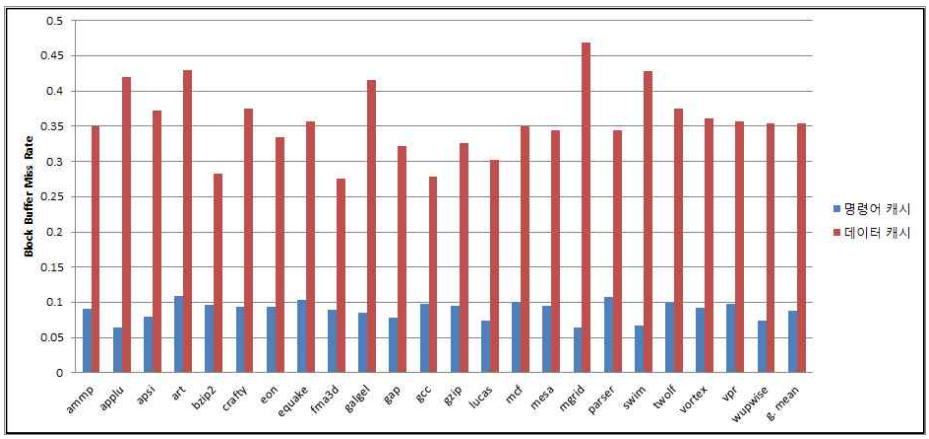

그림 5.7 명령어 캐시와 데이터 캐시에서의 필터 캐시 적중 실패율

그림 5.8은 이와 같은 적중 실패율을 가지고 있을 때 필터 캐시에 의한 성능 증가율을 분석한 그래프이다. CPI(Clock Per Instruction)를 측정하였으며 블록 단위의 필터 캐시 사용 시에 CPI의 감소율은 평균 64.7%이었고 같은 크기의 워드 단위 필터 캐시를 사용 시 평균 61.5%로 약 3.2%의 추가 감소가 있었다. 이는 동일한 크기에 저장 단위가 작아졌기 때문에 더 다양한 종류의 데이터들이 적재될 수 있어서 발생한 성능 향상으로 추정된다.

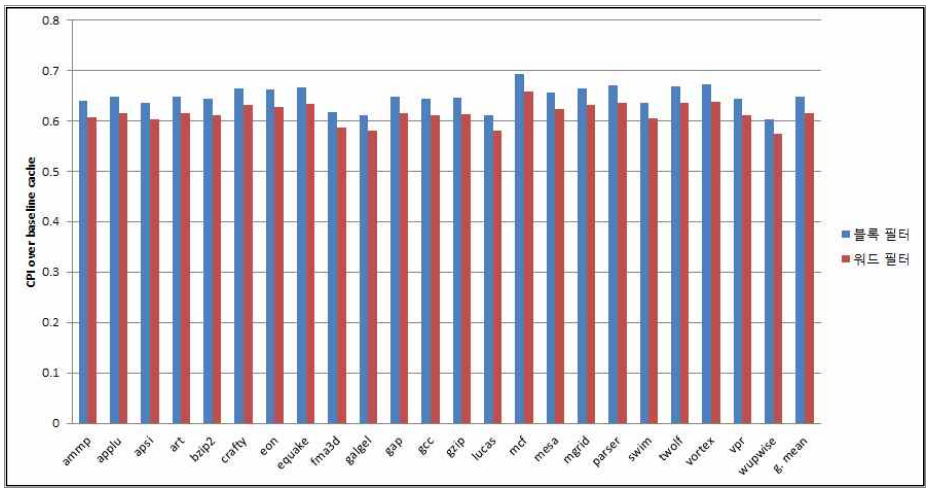

그림 5.8 블록 필터 캐시와 워드 필터 캐시에 의한 성능 증가율

5.3.2 드라우지 캐시에 의한 영향 분석

L1 데이터 어레이에 드라우지 캐시를 적용시켰을 때의 인자 값은 표 5.4와 같다.

표 5.4 드라우지 에너지 인자 값

누설 에너지 (per bit, per cycle)	드라우지 누설 에너지 (per bit, per cycle)	모드 변환에 드는 에너지
0.00163 pJ	0.000259 pJ	0.0256 nJ

2000 싸이클의 윈도우 크기를 가지는 드라우지 캐시를 구현했을 때, 전체 시간 * 공간 중 드라우지 모드에 있는 시간 * 공간 합의 비율에 대한 그래프가 그림 5.8이다. 블록 단위 관리 했을 경우 평균 87.5%, 워드 단위 관리 했을 경우 평균 90.0%로 2.9% 늘었다.

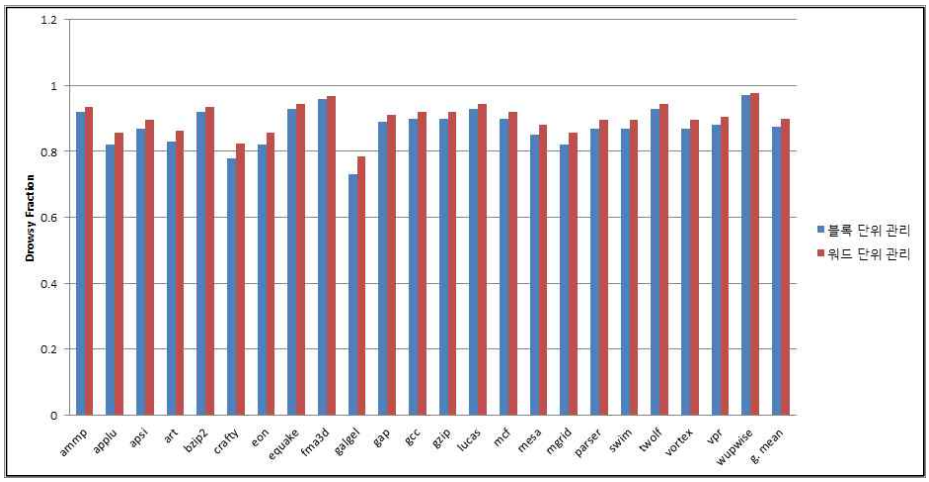

그림 5.9 블록 단위와 워드 단위로 관리했을 때 전체 시간 중 드라우지 모드에 있는 시간의 비율

5.3.3 SSDF 캐시의 에너지 소모량

5.3.3.1 동적 에너지 소모량

이론적으로 기준 캐시에 대해 성능 감소를 고려하지 않는다면, 필터 캐시를 구현했을 때 35% 정도의 동적 에너지 감소를, 순차적 캐시를 구현했을 때 60% 정도의 동적 에너지 감소를, 선택적 워드 접근 기법을 구현했을 때 67% 정도의 동적 에너지 감소를 관찰할 수 있다.

SSDF 캐시는 이들 모두를 구현하였지만 단순하게 이들의 동적 에너지 감소율을 곱한 만큼의 에너지 감소량이 나오지는 않는다. 필터 캐시에 적중이 될 경우, 순차적 캐시나 선택적 워드 접근 기법은 적용되지 않기 때문에 그들의 동적 에너지 감소 메커니즘이 효력을 발생시킬 수 없다. 순차적 캐시의 효력은 필터 캐시에 적중 실패가 발생했을 때 나타난다. 필터 캐시 적중 실패 후, L1 태그 어레이에도 적중 실패했을 경우 L1 데이터 어레이를 접근하지 않게 하여 그만큼의 동적 에너지 소모를 막을 수 있다. 또, 필터 캐시 적중 실패 후, L1 태그 어레이에 적중했을 경우 L1 데이터 어레이에 해당 웨이만을 활성화하여 동적 에너지 소모를 줄일 수 있고, 선택적 워드 접근 기법이 추가 적용되어 해당 워드만을 활성화할 수 있다.

순차적 캐시와 선택적 워드 접근 기법은 동시에 적용은 가능하지만, 성능의 보존을 위해 태그 어레이로 병렬 접근을 했기 때문에 오

버헤드가 발생한다. 그리고 드라우지 캐시를 구현하기 위한 오버헤드까지 고려하면 동적 에너지 감소량은 더 줄어들게 된다.

SSDF 캐시의 에너지 소모는 기준 캐시에 대해 평균 26.61%의 동적 에너지를 소모하는 것을 관찰할 수 있었으며, 그림 5.10 에 표현되어 있는 바와 같이 SWR 캐시에 비해 81.97% 만큼의 에너지를 소모하는 것을 관찰할 수 있다. CFP(Floating Point) 벤치마크인 Art와 Apsi, mcf와 같은 프로그램에서 성능이 좋지 않은 것을 관찰할 수 있으나 모든 벤치마크에서 SWR 캐시보다 동적 에너지가 적게 소모하는 것을 관찰할 수 있다.

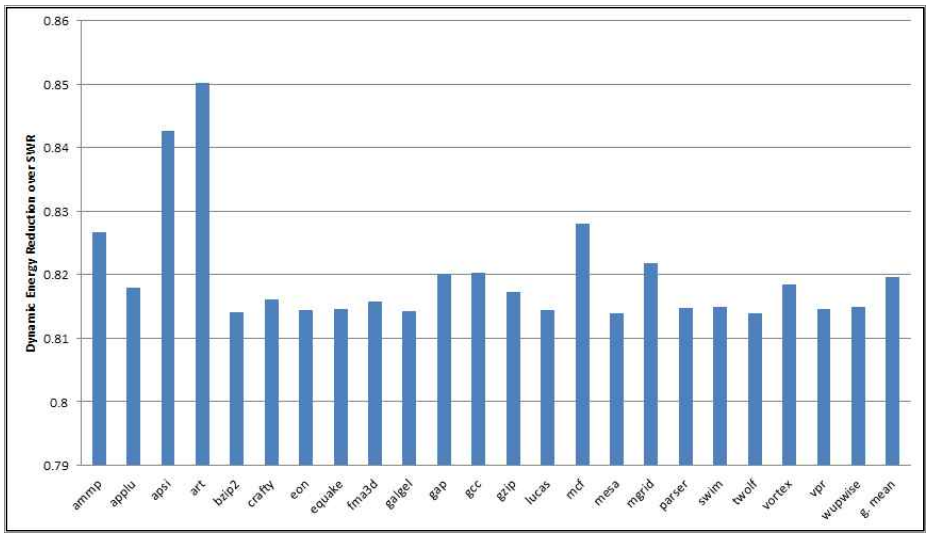

그림 5.10 SWR 대비 SSDF 캐시의 동적 에너지 소모량

5.3.3.2 정적 에너지 소모량

SWR 캐시는 기준 캐시에 대해 추가적인 하드웨어의 설치 같은 구조 변경이 없기 때문에 정적 에너지 소모량의 차이가 없다. SSDF 캐시에서는 필터 캐시 구조가 추가되었기 때문에 그에 해당하는 정적 에너지 소모량이 늘어나고, 그 외 추가적인 비트의 구현이 있기 때문에 오버헤드가 발생한다. 하지만, 드라우지 캐시가 구현되어 있기 때문에 전체적인 정적 에너지 소모량은 그림 5.11 과 같이 기준 캐시에 대해 평균 16.8%로 감소하는 것을 볼 수 있다. 5.3.2 절에서 살펴본 결과에 의하면, 드라우지 모드에 있는 시간 비율이 galgel, eon, crafty에서 낮고 fma3d, wupwise 등에서 높은 것을 볼 수 있으나, 프로그램 수행시간과 추가된 비트 등에 의해 정적 에너지 소모율의 그래프는 드라우지 모드에 있는 시간 비율 그래프와 큰 유사점이 없는 것으로 관찰되었다.

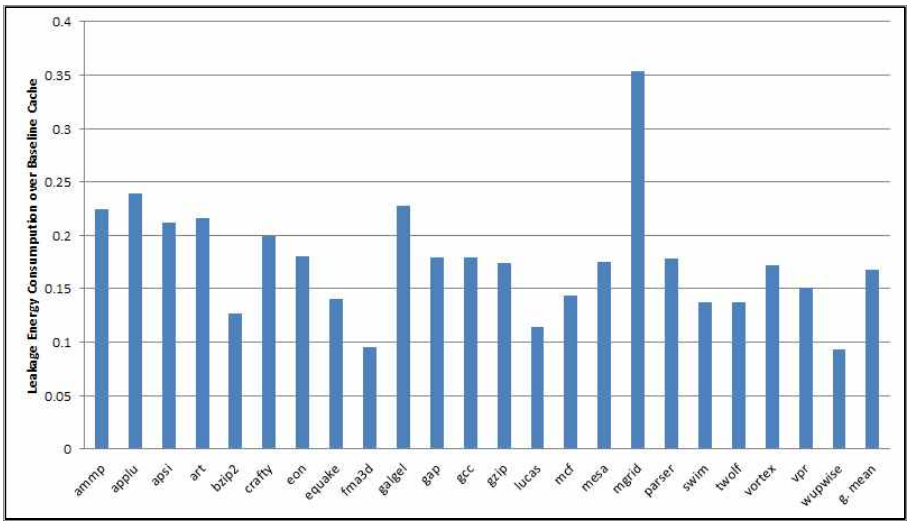

그림 5.11 기준캐시 대비 SSDF 캐시의 정적 에너지 소모량

5.3.3.3 전체 에너지 소모량

그림 5.12는 SWR 캐시에 대비하여 SSDF 캐시의 전체 에너지 소모량을 나타내는 그래프이다. SWR 캐시에 대해 평균 65.3%, Baseline에 대해 평균 28.3% 소모한 것을 확인해 볼 수 있으며, 그림 5.10과 그림 5.11에 비해 편차가 적은 그래프가 나타나는 것을 볼 수 있다. 이는 5.3.3.1에서 살펴본 동적 에너지 소모가 적은 벤치마크들은 필터 캐시 적중률이 높고, 그로 인해 실행시간이 빨라져 정적 에너지의 비율이 적어지기 때문이다. 반대로 동적 에너지 감소량이 상대적으로 적은 “Art” 같은 경우 실행시간이 상당히 길고, 그로 인해 정적 에너지 감소량이 반영이 많이 되어 전체적인 에너지 소모량이 고르게 분포하는 것을 볼 수 있다.

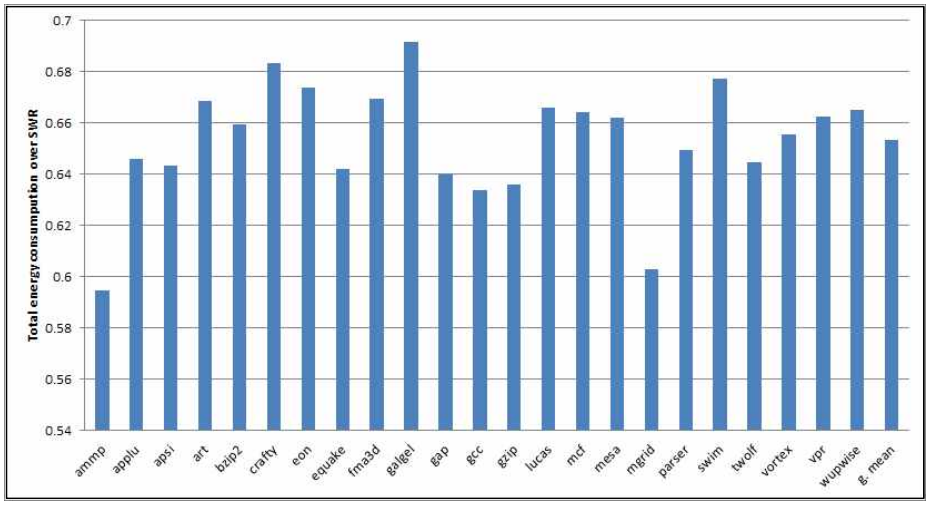

그림 5.12 SWR 대비 SSDF 캐시의 전체 에너지 소모량

5.3.3.4 비대칭 SSDF 캐시

3.4 절에서 논의한 바와 같이 명령어 캐시에서 워드 단위의 접근 기법을 사용할 경우 명령어 인출 너비의 변화로 성능이 감소할 수 있다는 것을 알 수 있었다. SSDF 캐시에도 선택적 워드 접근 기법을 적용하기 때문에 같은 현상이 발생한다. 명령어 캐시가 아닌 데이터 캐시에서는 공간적 연관성이 작기 때문에, 워드 단위의 접근이 성능 감소에 큰 영향이 없다는 것을 볼 수 있었고, 5.3.1 절에서 분석한 바와 같이 필터 캐시가 같이 구현되어 있는 경우 워드 단위의 접근이 더 효율적이라는 사실을 관찰할 수 있었다. 따라서 명령어 캐시와 데이터 캐시에서 접근 단위에 대해 비대칭적인 구조를 생각해볼 수 있다.

필터 캐시와 선택적 워드 접근 기법이 같이 구현되어 있는 SSDF 캐시의 경우, 접근 단위의 변화에 의해 성능 감소가 큰 명령어 캐시에는 2 워드나 4 워드 단위로 선택을 하고, 데이터 캐시에는 1워드 단위로 선택을 하는 혼합구조를 가정해볼 수 있다. 그림 4.13 은 해당 알고리즘을 사용했을 경우 발생하는 IPC를 관측한 그래프인데, 여기에서 ifq2와 ifq4는 모든 캐시에서 1 워드(명령어 인출 너비가 2, ifq2)나 2 워드(명령어 인출 너비가 4, ifq4) 단위로 선택을 하는 케이스이고, mix_ifq4와 mix_ifq8은 명령어 캐시에서는 2 워드(명령어 인출 너비가 4, mix_ifq4)나 4 워드(명령어 인출 너비가 8, mix_ifq8)인 케이스이다. 그래프는 블록 단위의 필터 캐시에 대비한 IPC의 변화를 관측한 것으로 이것은 ifq8로 명명한다.

명령어 인출 너비가 2이거나 4인 경우의 성능 감소는 이미 3.4절에서 관찰한 바 있으나, 해당 단위로 필터 캐시까지 구현된 경우를 관찰한 그래프가 그림 4.13 이다. 명령어 인출 너비가 2이고 필터 캐시의 단위가 1 워드인 경우 약 22% 가량의 IPC 감소가 발생했고, 명령어 인출 너비가 4이고 필터 캐시의 단위가 2 워드인 경우 약 6% 정도의 IPC 감소가 발생했다. 전술한 비대칭 구조의 경우, mix_ifq4인 경우는 ifq4인 경우와 비슷하게 약 5.5% 정도의 IPC 감소가 발생했으나 mix_ifq8인 경우에 데이터 필터 캐시에서의 적중률 증가로 오히려 1% 가량 IPC가 증가하는 것을 관찰할 수 있다.

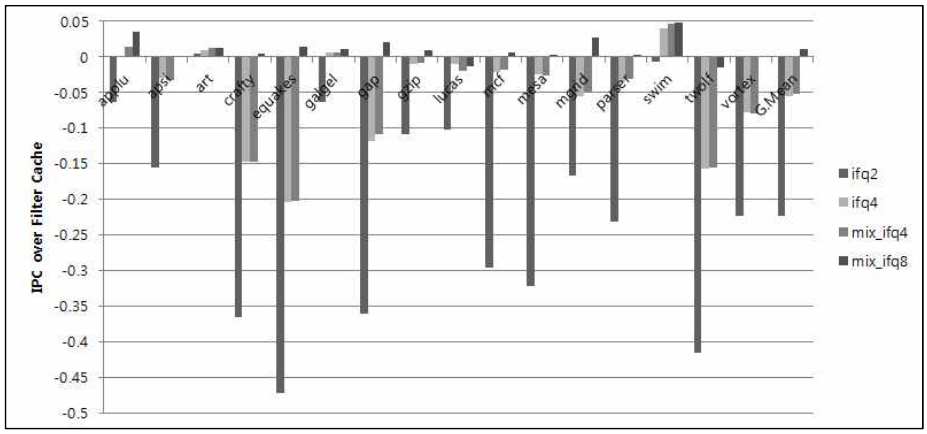

그림 4.13 비대칭 SSDF 캐시를 사용할 경우 필터 캐시 대비 IPC

필터 캐시의 단위의 변화와 비대칭 구조의 사용으로 인한 동적 에너지 변화를 관측하면 그림 4.14 와 같다. 다른 인자를 배재하기 위하여 선택적 접근 기법은 배재하고 실험을 진행했으며, 그로 인해 mix_ifq8인 경우 ifq8에 대비해 동적 에너지 소모량의 차이가 거의 없는 것을 관찰할 수 있었다. 두 알고리즘은 데이터 캐시에서 구성된 필터 캐시의 단위를 제외하고 모든 구조가 동일하며, 이는 IPC의 1% 증가와 99.99%의 동적 에너지 소모량을 보여주고 있다. 이 외의 ifq2인 경우에는 88.4%, ifq4인 경우 89.8%의 동적 에너지 소모량을 보여주고 있으며 비대칭 구조인 mix_ifq4인 경우 88.5%의 동적 에너지 소모량을 보여주고 있다.

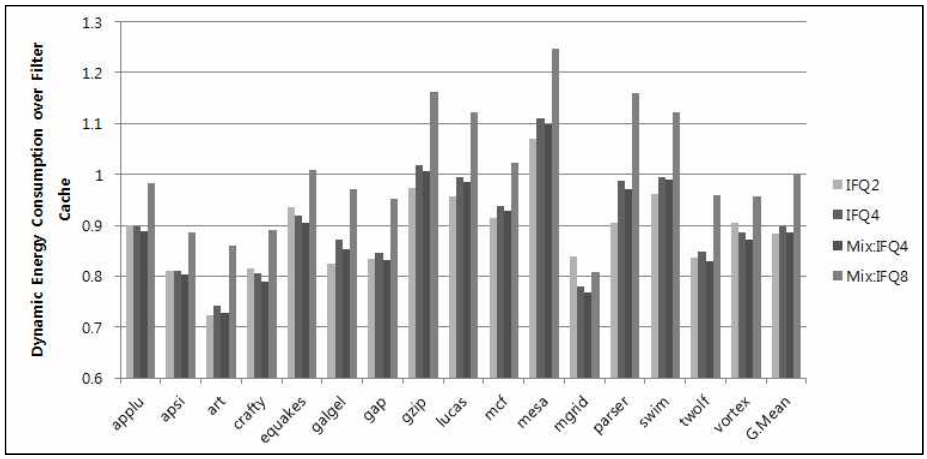

그림 4.14 비대칭 구조를 사용할 경우 필터 캐시 대비 동적 에너지

제 6장 결론

마이크로프로세서는 수행 성능을 증가시키고 소모하는 에너지를 줄이기 위해 연구가 진행되고 있다. 대부분의 경우 수행 성능과 소모 에너지들 간에는 트레이드오프(trade-off) 관계가 성립하여, 소모 에너지를 감소시키면 수행 성능이 낮아지게 된다. 본 논문에서는 프로세서의 구조적 개선을 통해, 수행 성능에 영향을 미치지 않으면서 소모 에너지를 감소시키는 방안과 수행 성능에 큰 영향을 미치는 여러 에너지 감소 방안들을 오버헤드를 최소화하면서 조합하는 방안을 제안한다.

첫 번째로, 수행 성능에 영향을 미치지 않으며 동적 에너지를 감소시키기 위해 '선택적 워드 접근 기법'을 제안한다. 저장장치 별 저장 단위가 다르다는 점에 착안한 이 기법은 주소의 일부분을 캐시 접근 시에 활용하여 저장장치 별로 필요한 부분만을 전달한다. 이 기법을 모의 실험하여 L1 캐시에서 67.5%, L2 캐시에서 27.1%의 동적 에너지 감소를 이끌어 냈다. 정적 에너지까지 고려하면 L1 캐시에서 56.75%의 에너지 감소를 이끌어 냈다.

두 번째로, 수행 성능에 큰 영향을 미치는 필터 캐시, 순차적 캐시 그리고 드라우지 캐시와 논문 전반부에서 제시한 선택적 워드 접근 기법을, 오버헤드를 최소화하면서 조합하는 '워드 필터를 사용한 순차적, 선택적 워드 접근 드라우지 캐시'를 제안한다.

필터 캐시는 프로세서 레지스터와 L1 캐시 사이에 작은 저장장치를 구현하여 동적 에너지 소모량을 줄이는 기법이다. 해당 기법이 처음 제시되었을 때와 달리 클록 수의 증가로 인해 L1 캐시 접근 시간이 늘어나고, 이로 인해 필터 캐시를 사용할 경우 에너지의 감소와 함께 성능상의 이득까지 볼 수 있다. 이와 함께 기존에 성능상의 손해로 인해 쓰지 못했던 순차적 캐시와 드라우지 캐시와 같은 기법들을 추가적으로 사용할 수 있다.

순차적 캐시는 캐시의 태그 어레이의 적중 여부를 알기 전까지 데이터 어레이를 동작시키지 않는 기법이다. 이는 태그 어레이의 적중 시간만큼 캐시 접근 시간이 길어지는 반면, 적중된 웨이만을 구동시키면 되기 때문에 데이터 어레이의 동적 에너지를 감소시킬 수 있다. 필터 캐시와 같이 사용할 경우, 상대적으로 전력 소모가 적은 태그 어레이를 필터 캐시와 병렬적으로 접근하게 되면 기존 순차적 캐시에서 손해를 보는 태그 어레이 접속 시간을 숨길 수 있다.

드라우지 캐시는 SRAM 셀에 동작전압을 정상 모드(높은 전압)와 저전력 모드(낮은 전압), 두 종류를 공급하고 동작이 발생하지 않는 부분의 전압을 낮추어 공급함으로 캐시의 정적 전력 소모를 줄이는 기법이다. 저전력 모드에 있는 셀에 접근할 경우 낮은 전압을 높은 전압으로 바꾸어주는데 이때 추가적인 접근 시간이 발생한다. 본 논문에서는 해당 셀에 접근하여 전압을 높이는 깨움 비트 전송을

필터 캐시와 L1 캐시 태그 어레이 접속과 병렬적으로 하여 기존 드라우지 캐시에서 발생하게 되는 성능 감소를 막았다.

이와 같이 드라우지 캐시 기법과, 필터 캐시, 순차적 캐시와 선택적 워드 접근 기법을 모두 적용하여 모의 실험한 결과, 전체 프로세서 캐시에서 73.4%의 동적 에너지 감소를, 83.2%의 정적 에너지 감소를, 총 71.7%의 에너지 감소를 이끌어 내었다.

요약하면, 정적 에너지 감소를 위해 드라우지 캐시를 구현하면서 발생하는 추가 시간을 필터 캐시와 순차적 캐시를 이용해 효율적으로 숨기고, 저장 단위 차이를 이용하는 선택적 워드 접근 기법을 추가적으로 구현해 저전력 프로세서 설계를 하였다.

참고 문헌

[ACC+09] J. Abella, J. Carretero, P. Chaparro, X. Vera and A. Gonalez, “Low Vccmin fault-tolerant cache with highly predictable performance”, *IEEE Micro*, 2009, pp. 111-121.

[Albo99] D. Albonesi, “Selective cache ways: on-demand cache resource allocation”, *International Symposium on Micro-architecture*, 1999, pp. 248-259.

[BTM00] D. Brooks, V. Tiwari and M. Martonosi, “Wattch: a framework for architectural-level power analysis and optimizations”, *International Symposium on Computer Architecture*, 2000, pp. 83-94.

[BV01] B. Batson and T.N. Vijaykumar, ”Reactive-associative caches”, *International Conference on Parallel Architectures and Compilation Techniques*, 2001, pp. 49-60.

[CRL03] Y. Chang, S. Ruan and F. Lai, “Design and analysis of low-power cache using two-level filter scheme”,.*IEEE Transactions on Very Large Scale Integration Systems*, 11(4), 2003, pp. 568-580.

[CGE96] B. Cadler, D. Grunwald and J. Emer, “Predictive sequential associative cache”, *International Symposiumon High Performance Computer Architecture*, 1996, pp. 244-253.

[CACT] Cacti simulator, http://www.hpl.hp.com/research/cacti/

[CL05] Y. Chang and F. Lai, “Dynamic zero-sensitivity scheme for low-power cache memories”, *IEEE Micro*, 25(4), 2005, pp. 20-32.

[FKM+02] K. Flautner, N.S. Kim, S. Martin, D. Blaauw and T. Mudge, “Drowsy Caches: Simple Techniques for Reducing Leakage Power”, *International Symposium on Computer Architecture*, 2002.

[GB+08] R. Giorgi and P. Bennati, "Filtering Drowsy Instruction Cache to Achieve Better Efficiency", *Proceedings of the 2008 ACM Symposium on Applied Computing*, 2008.

[GK99] K. Ghose and M.B. Kamble, “Reducing power in superscalar processor caches using sub-banking, multiple line buffers and bit-line segmentation”, *International Symposium on Low Power Electronics and Design*, 1999, pp. 306-315.

[GSG02] E. Gibert, J. S´anchez and A. Gonz´alez, “Effective Instruction Scheduling Techniques for an Interleaved Cache Clustered VLIW Processor”, *IEEE Micro*, 2002, pp. 123-133.

[HKY+95] A. Hasegawa, I. Kawasaki, K. Yamada, S. Yoshioka, S. Kawasaki and P. Biswas, "SH3: high code density, low power", *IEEE Micro*, pp. 11-19, 1995.

[HP06] L. Hennessey and D.A. Patterson, “Computer Architecture: A Quantitative Approach”, 4th Edition, Elsevier Science & Technology Books, 2006.

[HRY+01] M. Huang, J. Renau, S. Yoo and J. Torrellas, “L1 data cache decomposition for energy efficiency”, *International Symposium on Low Power Electronics and Design*, 2001, pp. 10-15.

[IIM99] K. Inoue, T. Ishihara and K. Murakami, “Way-predicting set-associative cache for high performance and low energy consumption”, *International Symposium on Low Power Electronics and Design*, 1999, pp. 273-275.

[INTE1] Intel Hompage Pentium 4 Specifications, *http://ark.intel.com/Product.aspx?id=27510&processor=&spec-codes=*

[INTE2] Intel Homepage Xeon 5000 Series Specifications, *http://www.intel.com/products/server/processor/xeon5000/index.htm#specifications*

[INTE3] Intel Homepage Xeon E7 Series Specification, *http://www.intel.com/products/server/processor/xeonE7/index.htm#specifications*

[JPP+04] P. Jung-Wook, G. Park, S. Park and S. Kim, “Power-aware deterministic block allocation for low-power way-selective cache structure”, *International Conference on Computer Design*, 2004, pp. 42-47.

[KBK02] C. Kim, D. Burger and S. W. Keckler, “An Adaptive, Non Uniform Cache Structure for Wire Delay Dominated On Chip Caches”, ASPLOS, 2002, pp. 211-222.

[KFB+02] N.S. Kim, K. Flautner, D. Blaauw and T. Mudge, “Drowsy Instruction Caches: Leakage Power Reduction Using Dynamic Voltage Scaling and Cache Sub-bank Prediction”, *International Symposium on Micro-architecture*, 2002.

[KFB+04] N.S. Kim, K. Flautner, D. Blaauw, T. Mudge, “Circuit and Microarchitectural Techniques for Reducing Cache Leakage Power”, *IEEE Transaction on Very Large Scale Integration Systems*, vol. 12, no.2, 2004.

[KGM97] J. Kin, M. Gupta, and W. H. Mangione-Smith, "The Filter Cache: An Energy Efficient Memory Structure", *Proceedings International Symposium on Micro-architecture*, 1997, pp. 184-193.

[KHM01] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting generational behavior to reduce cache leakage power", *Proceedings of the 28th International Symposium on Computer Architecture*, 2001.

[KIM07] R. Komiya, K. Inoue and K. Murakami, "Dynamic Management Technique to Mitigate Performance Degradation for Low-Leakage Caches", *Proceedings IEEE Symposium on Low-Power and High-Speed Chips: Cool Chips X*, 2007.

[KKS+10] C. Kozyrakis, A.Kansal, S. Sankar and K. Vaid, "Server Engineering Insights for Large-Scale Online Services", *IEEE Micro*, 2010, 30(4), pp. 8-19.

[KM08] T. V. Kalyan and M. Mutyam, "Word-Interleaved Cache: An Energy efficient Data Cache Architecture", *International Symposium on Low Power Electronics and Design*, 2008, pp. 265-270.

[KTK99] I. Koji, I. Tohru Ishihara and M. Kazuaki, "Way-Predicting Set-Associative Cache for High Performance and Low Energy Consumption," *International Symposium on Low Power Electronics and Design*, 1999.

[Lee11] 이정훈, "저전력 캐시를 위한 웨이-라인 예측 유닛을 이용한 새로운 드라우지 캐싱 기법", *정보통신설비학회 논문지,* 제 10권 제 2호, 2011, pp. 74-79

[LKT+02] L. Li, I. Kadayif, Y-F. Tsai, N. Vijaykrishnan, M. Kandemir, M.J.Irwin and A. Sivasubramaniam, "Leakage Energy Management in Cache Hierarchies", *International Conference on Parallel Architectures and Compilation Techniques*, 2002.

[LWK05] J. Lee, C. Weems, and D. Kim,"Selective block buffering TLB system for embedded processors", *IEEE Proceedings Computers and Digital Techniques*, vol. 152, no. 4, 2005, pp. 507-516.

[MRM03] G. Memik, G. Reinman and W.H. Mangione-Smith, "Reducing energy and delay using efficient victim caches", *International Symposium on Low Power Electronics and Design*, 2003, pp. 262-265.

[MWA+96] J. Montanaro, R.T. Witek, K. Anne, A.J. Black, E.M. Cooper, D.W. Dobberpuhl, P.M. Donahue, J. Eno, W. Hoeppner, D. Kruckemyer, T.H. Lee, P.C.M. Lin, L. Madden, D. Murray, M.H. Pearce, S. Santhanam, K.J. Snyder, R. Stehpany and S.C. Thierauf, "A 160 MHz, 32b 0.5W CMOS RISC microprocessor", *International Solid-State Circuits Conference*, 1996, pp. 214-215.

[NVN03] D. Nicolaescu, A. Veidenbaum and A. Nicolau, “Reducing data cache energy consumption via cached load/store queue”, *International Symposium on Low Power Electronics and Design*, 2003, pp. 252-257.

[Park11] 박성배, "Exascale 프로세서 기술 분석 및 전망", *전자공학회지 특집편*, 제 38권 제 5호, 2011, pp365-376

[PO04] P. Petrov and A. Orailoglu, “Tag compression for low power in dynamically customizable embedded processors”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(7), 2004, pp. 1031-1047.

[PO07] P. Petrov and A. Orailoglu, "Dynamic Tag Reduction for Low-Power Caches in Embedded Systems with Virtual Memory", *International Journal of Parallel Programming*, vol. 35, no. 2, 2007, pp. 57-177.

[PSS+05] S. Petit, J. Sahuquillo, J.M. Such and D. Kaeli, “Exploiting Temporal Locality in Drowsy Cache Policies,” *Computing Frontiers*, 2005.

[PYF+00] M. Powell, S. Yang, B. Falsafi, K. Roy, and T. Vijaykumar, "Gated-vdd: A circuit technique to reduce leakage in deep-submicron cache memories", *Proceedings of the International Symposium on Low Power Electronics and Design*, 2000.

[SD95] C. Su and A. Despain, “Cache design tradeoffs for power and performance optimizations: a case study”, *International Symposium on Low Power Electronics and Design*, 1995, pp. 63-68.

[Sega01] S. Segars, "Low power design techniques for Microprocessor", *Proceedings International Solid-State Circuits Conference Tutorial*, 2001.

[Simp] SimpleScalar toolset, http://www.simplescalar.com.

[Soda11] A. Sodani, "Race to Exascale: Opportunities and Challenges", *Keynotes International Symposium on Microarchitecture,* 2011.

[SPEC00] SPEC 2000 Benchmark, http://www.spec.org.

[TMA+08] S. Thoziyoor, N. Muralimanohar, J.H.Ahn and N.P.Jouppi, "CACTI 5.1", Technical Report, HP Laboratories Palo Alto, Apr. 2008.

[TMJ07] S. Thoziyoor, N. Muralimanohar and N.P.Jouppi, "CACTI 5.0", Technical Report, HP Laboratories Palo Alto, Oct. 2007.

[TTJ06] D. Tarjan, S. Thoziyoor and N. Jouppi, "CACTI 4.0: An Integrated Cache Timing, Power and Area Model", Technical report, HP Laboratories Palo Alto, June 2006.

[YPF+01] S.H. Yang, M.D. Powell, B. Falsafi, K. Roy and T.N. Vijaykumar, "An Integrated Circuit/Architecture Approach to Reducing Leakage in Deep-Submicron High-Performance I-Caches", *International Symposium on High-Performance Computer Architecture*, 2001.

[ZVY+05] C. Zhang, F. Vahid, J. Yang and W. Najjar, "A way-halting cache for low-energy high-performance systems", *ACM Transactions on Architecture and Code Optimization*, 2005, pp. 34-54.

[ZCZ07] M. Zhang, X. Chang and G. Zhang, "Reducing cache energy consumption by tag encoding in embedded processors", International Symposium on Low Power Electronics and Design, 2007, pp. 367-370.

[ZZ02] Z. Zhang and X. Zhang, "Access-mode predictions for low-power cache design", *IEEE Micro*, 22(2), 2002, pp. 58-71.

[ZZT+08] J. Zushi, , G. Zeng, H. Tomiyama, H. Takada and K. Inoue, "Improved Policies for Drowsy Caches in Embedded Processors", *Proceedings IEEE International Symposium on Electronic Design*, Test and Applications, 2008.

Abstract

The microprocessor is researched to improve the execution performance and reduce the energy consumption. In most cases, the trade-off relationship is established between the energy consumption and execution performance. So if reducing the energy consumption, the execution performance is lowered. In this paper, I propose two low power method by improving the architecture of the processor cache. The one is the method lowering dynamic energy without affecting the execution performance, and the other is the method combined some energy reduction plans which affect a significant impact on execution performance.

First, I propose 'Selective Word Reading(SWR)' technique which reduce the dynamic energy of the processor cache without loss of performance. This technique was developed because of the differences between store unit sizes per storage level. In the SWR cache, only the necessary part of a block is activated during the cache access process. For a 32 kB four-way set associative L1 cache, a 32B block size, and four mats per sub-bank, the SWR cache saves dynamic energy by 67.5% without consideration of the leakage energy and by 56.75% with consideration of the leakage energy with no performance degradation and negligible area reduction. Additionally, in a 1MB 16-way set associative L2 cache, a 64B block size, and eight mats per sub-bank,

the SWR cache saves dynamic energy by 27.1% for the L2 cache.

Second, I propose ‘Sequential-SWR-Drowsy Cache with the Word Filter(SSDF)’ technique which reduce the entire energy of the processor cache with combining a sequential cache, a selective word reading, a filter cache and a drowsy cache. These techniques are affecting a significant impact on execution performance and I offer the method which can reduce the performance overhead with maximizing the effect of the power consumption.

The filter cache is a technique to reduce the dynamic energy consumption that implements a small storage device between the L1 cache and the processor registers. Unlike when it is presented first, by increasing the number of CPU clocks, the access time of the L1 cache is increased and thus, the filter cache, this approach can be seen to advantage of the performance as well as the power consumption. Furthermore, it is possible to use further techniques such as the drowsy cache and the sequential cache without additional damage to the performance.

The sequential cache is a technique to delay the operation of the data array until the tag array knows whether it is hit or not. Since the access time of the sequential cache is increased by the tag-array-access time, and to drive only the hit way, so it is possible to reduce the

dynamic energy of the data array. When used with the filter cache, if accessed in parallel with the filter cache and the L1 tag array whose power consumption is relatively small, it can hide the tag-array-access time.

The drowsy cache supplies the two kind of the operating voltage to the SRAM cell and it makes the cells is placed in two modes – normal mode in high voltage and drowsy mode (low-power mode) in low voltage. And the some cells which access rarely will be placed in drowsy mode, it will decrease the static energy consumption of the cache. If an application want to access the cell of the drowsy mode, at this time that it converts the low voltage to the high voltage, and it will make the additional access time. In this paper, we prevented the degradation of performance by the parallel access of the wake-up call is occurred when the filter cache and the L1 cache tag array is accessed.

This technique, SSDF cache, saves 73.4% of the dynamic energy, 83.2% of the static energy and 71.7% of the total cache energy consumption.

Keywords : Selective Word Reading, Filter Cache, Sequential Cache, Drowsy Cache, Parallel Request Structure, Low-power Cache Architecture